The Knowledge Engineering Review

The control of reasoning under uncertainty: A discussion of some programs*

PAUL R. COHEN

Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003†

Abstract

This paper proposes that managing uncertainty is a *control* problem, a task for the control component of AI systems that decides what to do next. This view emphasizes the process of planning and executing sequences of actions that simultaneously satisfy domain goals and minimize uncertainty. The paper reviews AI systems that manage uncertainty by control. It is not an exhaustive survey, but rather illustrates issues in managing uncertainty with selected AI programs.

1 Introduction—managing uncertainty by control

This paper is about how problem solvers decide what to do when they are uncertain. For AI programs, these decisions are implemented by control strategies that determine the programs' focus of attention, and limit their inferences and evidence-gathering questions. Control strategies have been neglected in expert systems; there is a strong belief that the power of expert systems is due to substantive or inferential knowledge, and that simple algorithms such as forward or backward chaining suffice to control its application. But control strategies must be considered part of the expertise of many domains; for example, when we describe a doctor as an expert diagnostician we mean that he or she knows how to do diagnosis. This does not deny the role of substantive medical knowledge in diagnosis, but suggests that control strategies, which specify what to do in uncertain situations, should be accorded the same status as other kinds of knowledge. This paper describes selected programs that treat control strategies as a kind of expertise, focusing especially on programs that were developed for uncertain task domains.

The paper begins with an overview of managing uncertainty by control. Section 2 defines the relationship between expert problem-solving strategies and their implementation in control strategies, and the difficulties this poses for knowledge engineers. Decision analysis and AI planning techniques are discussed in Section 3 as potential mechanisms for managing uncertainty by control. Along the way, a criterion of efficiency is developed for evaluating how well a problem solver manages uncertainty by control. In Section 4, the blackboard architecture is introduced and the Hearsay-II speech understanding system is discussed in detail. Recent efforts to make control in blackboard systems more explicit are surveyed. The description of Hearsay-II is extensive because its designers were perhaps the first to consciously rely on control strategies to manage uncertainty, and many of their ideas generalize beyond speech understanding. But while blackboard systems offer the most opportunity for sophisticated control, rule-based production systems are the most common architecture for expert systems. Accordingly, Section 5 discusses managing uncertainty by control in rule-based architectures. Section 6 describes work in the author's laboratory on planning efficient medical diagnoses. Section 7 is a discussion and conclusion. Section 8 offers a structured bibliography.

* The research was supported by DARPA-RADC Contract F30602-85-C0014 and NSF Grant IST-8409623.

† This paper results from ongoing research with David Day, Jeff DeLisio, Mike Greenberg, Tom Gruber, Adele Howe, Rick Kjeldsen, Dan Suthers and Dr. Paul Berman, M.D. I am grateful for fruitful discussions with Professor Victor Lesser and Professor Glenn Shafer.

Most AI research on reasoning under uncertainty is concerned with methods for combining evidence. There is much debate about which methods are appropriate and why. However, good strategies for reasoning under uncertainty should not depend on how one combines evidence and should be robust against violations of assumptions such as conditional independence. This paper therefore says nothing about the voluminous literature on combining evidence, other than to refer the reader to representative sources. (Shortliffe and Buchanan, 1984), (Shafer, 1976), (Shafer, 1984), (Duda, Hart, and Nilsson, 1976), (Pearl, Leal, Saleh, 1982), (Quinlan, 1983), (Lowrance and Garvey, 1983), (Gordon and Shortliffe, 1984), (Szolovits and Pauker, 1978), (Bonisonne, 1985), (Cohen and Gruber, 1985), (Fox, 1985) (Henrion, 1986), (Sullivan and Cohen, 1985), (Cohen and Lieberman, 1983), (Cohen, Shafer, and Shenoy, 1987), (Shafer and Tversky, 1986).

2 Managing uncertainty by control

Managing uncertainty is a style of problem solving that assigns to the problem solver the task of deciding what to do when it is uncertain. The word "managing" connotes an active role for the problem solver, which is responsible for achieving its goals despite uncertainty. The mechanisms for implementing this style of reasoning are collectively called the problem solver's *control strategy*. All AI systems have control strategies that dictate at least two and sometimes three aspects of behavior. Most interesting problems are solved by breaking them into subproblems, and often each of these can be achieved in several ways. Control strategies tell a system how to select a *focus of attention*—which piece of a problem to work on next. Control strategies also dictate *control of inference*, preventing a system from applying all its reasoning methods to all pieces of a problem. Focus of attention specifies a piece of a problem, control of inference specifies how to work on it. Additionally, some AI problem solvers can ask for information, or take actions to affect the state of the world, or in other ways interact with an external environment. These systems require a third aspect of control, namely *control of actions*.

These mechanisms are illustrated by an example from medical diagnosis:

A patient walks into the doctor's office with a high temperature. This finding is consistent with dozens if not hundreds of disease and ailments. The doctor must control the inferences that can be made from the finding of high temperature, or else be swamped by hypotheses about what is wrong with the patient. Control of inference implies that not all possible inferences are made. Of those that *are* made, the doctor will recognize one or two possibilities as especially likely, or serious, or otherwise worthy of attention. One will become the focus of attention, and the doctor will formulate a goal to confirm it or rule it out. Now, the doctor must decide which actions will obtain more information at an acceptable cost—which questions, tests, or treatments will provide needed evidence. Once obtained, evidence will enable further inferences, and perhaps a revised focus of attention and other goals.

The efficiency and success of uncertain processes like medical diagnosis depends on control strategies. If the focus of attention is too difficult to change, the process may "fixate" on the wrong disease; but if the focus is constantly changing, then diagnosis will never succeed. If too many inferences are made, the diagnostician will falter under a confusion of possibilities; if inference is restricted, then the diagnostician will miss diagnoses and perhaps misdiagnose. But whereas the cost of inference for AI programs is measured in machine cycles, the costs of actions are measured in human terms: too many questions, tests, and treatments will inconvenience and possibly harm the patient; too few, and the result will be similarly undesirable.

2.a Control strategies and problem-solving strategies

Control strategies implement problem-solving strategies. The distinction is not always clear-cut, but it has important consequences for knowledge engineering. A physician may describe his or her problem-solving strategy this way:

"First I take a history, which usually triggers a few possibilities. I try to narrow the differential as much as possible during the history. Next I do a physical exam. I'm looking for signs that help me to narrow the differential further. The physical can also help me refine my suspicions: The history may tell me there's a cardiac

problem, but the physical can narrow it to, say, mitral valve prolapse. I avoid tests, especially invasive ones, and rarely perform them except to confirm something I already believe pretty strongly from the history and physical."[1]

The astute knowledge engineer can recognize fragments of a control strategy in this description. Four kinds of actions—history, physical, and noninvasive and invasive tests—are distinguished. They appear to follow that order: first, questions about the chief complaint, past medical history, and so on; then the physical exam; then tests, beginning with those that are least risky and uncomfortable. The goal of the physical exam is to narrow the set of hypotheses from which the focus of attention is selected (the "differential"); though it may also add hypotheses to that set if they replace less specific hypotheses. The goal of tests is to confirm strongly believed hypotheses. Inference is controlled during the history; the only inferences permitted are those that "trigger" disease hypotheses.

The knowledge engineer is responsible for translating the physician's problem-solving strategy into a control strategy. Two related concepts will help us to understand this process: First, the physician and the knowledge engineer each have a vocabulary of *primitive* terms. For the expert these include familiar medical terms such as "trigger" and "differential"; for the knowledge engineer they are the mechanisms that implement focus of attention, control of inference, and control of action, such as "the agenda," and "conflict resolution" and "heuristic evaluation function." Second, there is a *distance* between the physician's vocabulary and that of the knowledge engineer that depends on the difficulty of translating one into the other.[2]

Historically, knowledge engineers have sought to reduce this distance by providing experts with a language of *task-level* primitives; at least, this has been the preferred strategy for acquiring inferential knowledge. A physician can say, "If a patient is female and under 20 years old and has no family history of heart disease, then I am sure she does not have heart disease," without worrying about how the conjunctive "and" is implemented, or how the implication is interpreted, or how degrees of belief are calculated. The physician can tell the knowledge engineer that mitral valve prolapse is a kind of heart disease, and neither of them need consider how the *kind-of* relation is implemented. The role of the knowledge engineer is to find the task-level primitive terms in the expert's vocabulary (e.g., disease, kind-of, and, if-then, ...), implement them, and then communicate with the expert in terms of the domain—task-level terms—without considering their implementation.

But historically, problem-solving strategies have not been acquired this way. They have been represented by *implementation-level* primitives such as the agenda and conflict resolution. Problem-solving strategies have been inferred, not acquired from experts. Adding or modifying a problem-solving strategy could take days or more of programming or manipulating low-level parameters. The distance between the task-level and implementation-level primitives in the vocabularies of strategy was very large.

This is beginning to change. Some researchers view problem-solving strategies as expertise. They seek to acquire strategies by the same methods as inferential knowledge; that is, by first implementing a set of task-level primitives with which experts can describe their strategies, (Gruber and Cohen, 1987b), (Hayes-Roth, Garvey, Johnson, and Hewett, 1986), (Clancey, 1986), (Chandrasekeran, 1986), (Bylander, 1986), (Marcus, 1985), (Kahn, 1985). Let us use the term *control feature* to refer to any task-level primitive term that is interpreted at the implementation level to discriminate alternative control decisions, including focus of attention, control of inferences, and control of actions. For example, if we want a medical expert system to focus attention on dangerous potential diseases, then we must associate a *dangerousness* control feature with every disease hypothesis, and tell the system to attend to those hypotheses for which the value of that feature is high. Similarly, we might want to control inferences by restricting the system to inferences that potentially increase our belief in hypotheses. This requires a control feature to represent the potential difference between the current and anticipated levels of belief. To control actions, we need to differentiate their features, such as cost, reliability, time requirements, and so on.

1 The author is indebted to Dr. Paul Berman, University of Massachusetts Health Service, for instruction on the process of diagnosis.

2 The following discussion borrows from Gruber and Cohen (1987a, 1987b).

This view has several advantages. As long as control strategies were built into the "inner loop" of knowledge systems, there could be no experimentation or modification without tearing the system apart. (This may explain why control is a neglected area of expert systems.) Task-level control features for representing strategies facilitate this kind of experimentation. They also enable flexible control, because systems with declarative control features can reason about their own control and decide which control strategies are most appropriate to particular situations. For example, a medical system should control diagnosis one way if the patient requires immediate attention and another if the patient is stable. Task-level control features also facilitate acquisition of problem-solving strategies from the experts who have developed them—far preferable to relying on knowledge engineers to invent strategies that they hope will solve problems in the same ways as experts.

Despite these advantages, the trend toward explicit representations of control knowledge is relatively recent. As we will see in the following sections, however, it plays an essential role in providing expert systems with the ability to manage uncertainty.

3 The relevance of problem- solving, decision analysis, and planning to managing uncertainty by control

In this section, I compare managing uncertainty by control with methods from AI and decision analysis. The goals of this comparison are to further elaborate what I mean by managing uncertainty by control, to examine the relevance of these established methods, and to borrow from these methods criteria for evaluating how well a system manages uncertainty by control.

3.a Search and problem-solving

Managing uncertainty by control means selecting actions that simultaneously achieve domain goals and reduce uncertainty. If we treat reducing uncertainty as just another goal, then we can describe managing uncertainty in conventional problem-solving terms: uncertainty gives rise to goals that are achieved by "uncertainty-reducing" actions. Actions reduce uncertainty in a variety of ways: all actions produce evidence, some "hedge" over possible outcomes, some minimize the worst-case outcome, and so on. This suggests that problem solvers act for two kinds of reasons: to achieve domain goals (e.g., diagnose a patient) and to achieve adequate certainty. Since all actions produce evidence, every action can be selected for both kinds of reasons.

Because actions are selected both for their domain effects and for their effects on uncertainty, problem solving under uncertainty is more constrained than reasoning under the assumption of certainty. Contrary to intuition, this implies that uncertainty may actually *facilitate* problem solving. Uncertainty provides a basis for focusing attention (e.g., focus on the hypothesis for which inexpensive evidence will produce the largest change in certainty), and also for selecting actions (e.g., take the action that will do the most to disconfirm the current focus). Three control features seem especially important if uncertainty is to affect problem solving. These are: the current degree of belief in a hypothesis, the prior probability of the hypothesis, and the potential degrees of belief in the hypothesis given some action. A problem solver may focus on the hypothesis that has the highest current degree of belief, or on the hypothesis that is most likely a priori, or perhaps on an hypothesis that is both a priori likely and currently credible. It may then select an action that potentially confirms the focus of attention, or potentially disconfirms it, or, ideally, does both depending on the outcome of the action. Through these and related control features, uncertainty can facilitate problem solving by constraining focus of attention and action selection.[3]

3.b Efficiency and decision analysis

Managing uncertainty by control typically involves tradeoffs between goals. If we assume that all evidence has costs, broadly construed, then managing uncertainty always involves a fundamental

3 This point is reminiscent of (Waltz, 1975), that extending the set of line types to include shadow regions did not impede but supported line-labelling of blocks-world images.

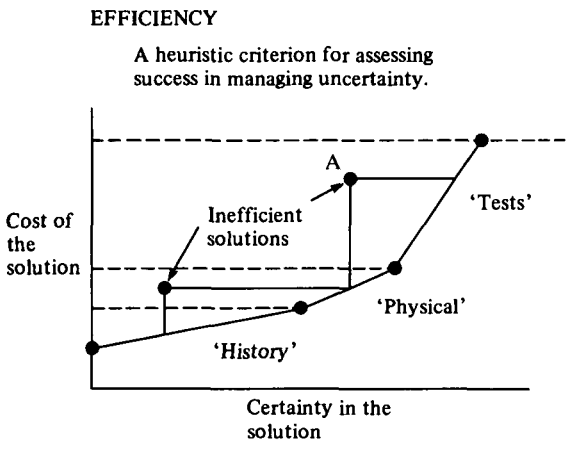

Figure 1 One inefficient solution incurs the costs associated with physical exam for no more certainty than could be had from the history

tradeoff between costs and certainty. It follows that problem solving can be more or less efficient with respect to this tradeoff. An efficient solution maximizes certainty for a given cost and minimizes cost for a given level of certainty. For reasons discussed shortly, efficiency is rarely a normative criterion; that is, one can rarely prove that a control strategy will produce the true maximum certainty for a given cost and the true minimum cost for a given level of certainty. But efficiency provides a good heuristic tool for evaluating control strategies. If a program could have avoided asking for superfluous evidence, then it is inefficient to some degree, and its control strategy might require modification.

The concept of efficiency is illustrated by the curve in Figure 1. Each point on the curve is efficient in the sense of providing maximum certainty for given costs and minimum costs for given levels of certainty. The curve has three regions corresponding to three phases of medical problem solving. The relatively flat region, labelled "history," corresponds to the initial phase of diagnosis, which includes questions about symptoms, past medical history, and so on. The steeper region corresponds to the physical exam and the steepest corresponds to tests such as EKG and angiogram. The history region has a relatively gradual slope because physicians can narrow their space of diagnoses substantially by asking inexpensive questions. A skilled physician knows which questions to ask to generate a small set of potential diagnoses, called the differential. Often, the questions are so efficient that the differential contains only one or two hypotheses. Depending on the diseases in the differential, the efficiency of the "physical exam" region may be high or low. In diagnosing heart disease, the physical is usually uninformative and could almost be skipped except that it occasionally finds unexpected results. The "test" phase of diagnosis is even less efficient, as indicated by its steeper slope in Figure 1. Many physicians feel that the role of tests is to confirm diagnoses that are all-but-certain from the history and physical. Given the relatively high costs of tests—both in terms of money, time, and discomfort—no physician wants to order them unless he or she has a pretty good idea of the diagnosis.

This is not to say that tests are worthless, only that they are less efficient than questions asked during the history phase of diagnosis. An inefficient diagnosis would be one in which tests were performed when they could not provide useful evidence. This is represented by point A in Figure 1. Here, the physician has incurred the cost of tests but achieved no more certainty than he or she had during the physical exam. In general, any point in the interior of the curve in Figure 1 is inefficient, because a point below it could provide equal certainty at a lower cost, and a point to the right could provide more certainty at the same cost.

The fundamental tradeoff between the cost of actions and the utility of the evidence they produce suggests that one might implement the management of uncertainty as a series of normative decisions

about actions, each conditioned on the current state of knowledge about the problem. Several proposals have been made to use decision analysis (Raiffa, 1970), (Howard, 1966) as a mechanism for selecting actions.[4] These seem unlikely to succeed, because decision analysis requires a *complete*, combinatorial model of each decision. The expected utility of each potential action can only be calculated from the joint probability distribution of the possible outcomes of the previous actions. The benefit of this exhaustive analysis is that problem solving is guaranteed efficient: One will always achieve maximum certainty for a given cost and minimum cost for a given level of certainty. The heuristic approaches discussed in this paper do not guarantee efficiency, but for many, the principle of efficiency underlies their design. (An alternative *constructive* approach to decision making is presented in Howe and Cohen (1986).)

3.c Planning

One form of the principle of efficiency says that a problem solver should never take an action that cannot possibly change its level of belief in any hypothesis. One often cannot tell in advance whether an action will *actually* change one's level of belief. This is because the current state of a problem may be incompletely known, and the effects of actions can only be estimated. For example, a detective may ask a question that *potentially* confirms the identity of a murderer, but he cannot know that it will actually do so unless he knows the answer in advance, in which case the question is superfluous. The principle of efficiency says that although the detective may be unable to ask a question that is guaranteed to confirm his suspicion, he should not ask a question that cannot potentially do so. Every action should potentially provide useful evidence. But this implies dependencies between actions, because the evidence that may be produced by one action can render another superfluous.

Dependencies between actions help the problem solver to order actions. This is planning, though not planning in the usual AI sense of the word (Sacerdoti, 1979), (Cohen and Feigenbaum, 1982). The differences are due to the fact that the state of a problem and the effects of actions are both uncertain. The uncertain planner must "feel its way" by estimating the likely outcomes of one or more actions, executing them, then checking whether the actual state of the world is as expected. Plans in uncertain environments tend to be short. In contrast, most AI planners excise uncertainty by assuming that the initial state of the world and the effects of all actions are completely known (e.g., the STRIPS assumption (Fikes, Hart, and Nilsson, 1972)). AI planners can proceed by "dead-reckoning", because it follows from these assumptions that every state of the world is completely known.

Dead-reckoning planning is unrealistic, but it is not irrelevant to managing uncertainty by control. Particularly germane is the idea of constraint-based planning (Stefik, 1980), originally developed to handle subgoal interactions. Before constraint-based planning, planners were quite capable of taking actions that negated the results of previous actions and destroyed the preconditions of subsequent actions. Constraint-based planners catch these problems before they arise and use them as a basis for ordering actions. For example, imagine a family with one car in which the wife needs to be at work before the husband, and the husband needs the car during the day. If constraints are not considered, then a plan might be developed that has the wife drop the husband at work and then go to her business. But this means the husband won't have the car later, and the wife arrives late. Bad as this planning is, it can get worse: Once the planner realizes its error, it might "fix" the plan by inserting another step: now both drive first to the husband's workplace, then both drive to the wife's, then the husband returns to his business. This plan achieves the goal but includes a wasted (and negated) step. The correct way to solve the problem is to reason from the constraints that the couple should drive first to the wife's business, then the husband should take the car to work.

In this example, constraints hold among actions in the physical world. But constraints also hold among actions that produce evidence, that is, actions that affect both the physical world and our knowledge about it. This is where we see the relevance of constraint-based planning to managing

[4] All are in unpublished reports; I know of none that has actually been implemented and published.

uncertainty. For example, imagine wanting to buy a birthday present and a box in which to wrap it. Which should you buy first? If you know ahead of time how big the birthday present will be, then the order of the actions is irrelevant. But if not, the knowledge provided by buying the present is a preconditon for buying the box; and this constraint dictates the order of the actions.

It might be possible to modify constraint-based dead-reckoning planners for managing uncertainty. The assumptions on which they are based—that the state of the world and the effects of actions are completely known—seem to deny uncertainty. However, if constraint-based planners considered a relatively small number of *potential* outcomes of actions (and thus potential states of the world), then they might prove useful despite the combinatorial increase in the size of their search spaces. The utility of this approach remains to be seen. We turn our attention now to approaches that rely on manipulating the control strategies of knowledge systems to manage uncertainty.

4 Strategic approaches

Control strategies tell AI programs what to do next. Earlier, the three components of control strategies—focus of attention, control of inference, and control of questions—were discussed. In this Section, we survey how these functions can result in efficient management of uncertainty; that is, solutions to domain problems that maximize certainty for given costs and minimize costs for given levels of certainty. Although this is a heuristic criterion, one can easily see how it is violated by poor control strategies. The following discussion focuses on approaches to control in four architectures: the Hearsay-II blackboard, meta-rules, NEOMYCIN, and an inference-network-plus-planner developed in the author's laboratory.

4.a Blackboard architectures and Hearsay-II

Blackboard architectures have the following structure: A set of *knowledge sources* monitors a common data structure called a blackboard. Roughly characterized, each knowledge source contains a set of production rules. The left-hand sides or conditions of the rules in a particular knowledge source "look" in the same place, or for the same kind of event, on the blackboard. For example, a knowledge source that contains rules for prescribing therapy may look for a small set of diagnoses to be posted on the blackboard. This event would tell the therapy knowledge source that the system has narrowed the differential enough to consider therapy. The right-hand sides or conclusions of the rules in knowledge sources post an event on the blackboard—such as adding a disease to the differential, or prescribing therapy—provided they are permitted to do so by the scheduler.

Typically, blackboards are divided on several dimensions. Time is an important dimension for many signal-interpretation tasks. Blackboards usually have a dimension that organizes *levels* that correspond to levels of analysis in a task; for example, a medical diagnosis system may have four levels: the *findings* level, containing signs, symptoms, and the results of tests; the physical *system* that is affected by disease (e.g., respiratory, cardiovascular, etc.); the *class* of diseases that can be inferred from the findings and the identified system (e.g., coronary artery problems); and the *specific disease hypotheses* (e.g., congenital narrowing of the coronary arteries). Knowledge sources typically look at one level of the blackboard for stimuli (events to match the left-hand sides of rules) and post conclusions either at the same level or at a different one. For example, a knowledge source that "triggers" specific disease hypotheses given particular clinical findings looks at the *findings* level for events that match its conditions and posts conclusions on the *specific disease hypotheses* level.

When a knowledge source sees an event that matches its conditions, it generates a request to the scheduler called a knowledge source instantiation (KSI). A KSI says, essentially, "I am a knowledge source who has something to contribute, if you will allow me to do so." The scheduler must then decide which KSI to honor. In simple blackboard systems, most KSIs propose to add something to the blackboard. Each addition can potentially stimulate other knowledge sources and generate other KSIs. Consequently, the scheduler cannot honor all KSIs, otherwise a combinatorially-increasing number of things will be posted on the blackboard. The scheduler must control KSI

execution in a way that ensures that a global solution evolves efficiently. Most research on blackboard systems is concerned, in one way or another, with this scheduling task.

The blackboard architecture was invented to manage the uncertainty inherent in complex signal-understanding tasks.[5] The Hearsay-II system, where blackboards were first used, interpreted connected speech (Erman, Hayes-Roth, Lesser and Reddy, 1980). This task involves massive uncertainty from several sources. Not only is the speech signal noisy, but the knowledge sources that interpret it are typically errorful. On the other hand, speech is redundant in the sense that constraints hold between sounds within a word and between words in a sentence. It is possible to exploit these constraints during interpretation of utterances. Knowledge sources (KSs) make inferences about what is possible given what is already believed. However, unrestricted application of knowledge sources to the speech signal generates an intractable number of fragments of interpretations:

> At any level of analysis, a very large number of errors may be introduced, including misclassifications, failures to recognize, and inappropriate "don't care" responses to truly significant portions of the utterance. The common approach in speech understanding research is to construct systems that can recognize utterances in spite of such errors by evaluating simultaneously many weakly supported interpretations of the speech. . . . One objective of *attentional control* is to schedule the numerous potential activities of the KSs to prevent the intractable combinatorial explosion that would inevitably result from an unconstrained application of KSs. More specifically, the focus of attention problem is to minimize the total number of KS executions (or total processing time) necessary to achieve an arbitrarily low rate of error in the semantic interpretation of utterances (Hayes-Roth and Lesser, 1977, p. 27).

The blackboard in Hearsay-II can be envisioned as a two-dimensional structure in which time extends along the horizontal axis and levels of analysis of the speech signal extend vertically. At the lowest level are structures that represent the spectral characteristics of the utterance. These extend from left to right in the order in which they are spoken. This temporal sequence is common to higher levels of analysis, too. Above the spectral representation is a segmented phonological representation, then levels representing syllables, words, sequences of words, and, at the highest level, phrases. Knowledge sources (KSs) combine objects at one level to produce hypotheses at a higher level; for example, the WORD-SEQ KS uses statistical knowledge about the frequency with which words are paired in English to produce word sequence hypotheses (at the word-sequence level) from combined word hypotheses at the word level. Knowledge sources also make predictions and ask for verification from "lower" in the blackboard. Some knowledge sources are used to implement control strategies.

Scheduling knowledge source instantiations (KSIs) in Hearsay-II was fairly complex and required significant empirical tuning. The designers of Hearsay-II experimented with several strategies, though one eventually proved better than the others. It had two components, executed in sequence, that we will call *fixed* and *opportunistic*, respectively. The fixed component first called knowledge sources to generate morphological interpretations of the parameterized speech signal, then generated some word-hypotheses. The most likely word hypotheses were left on the blackboard and the opportunistic phase of processing was initiated. During this phase, Hearsay-II scheduled KSIs at all levels of the blackboard according to the degree to which they potentially satisfied five criteria. The "best" KSI, according to the criteria, was executed, the results posted on the blackboard, and the process repeated.

The criteria used to schedule KSIs are interesting for their generality. They can be found in other blackboard systems as well as non-blackboard architectures (e.g., see the discussion of metarules later in this Section). They comprise a general set of tactics for managing uncertainty by control:

4.a.1 Efficiency

Reliable and inexpensive KSs should be executed before less reliable or more expensive ones.

[5] The range of tasks for which blackboard systems have been developed is nicely described in a two-part survey article by Nii (1986a, 1986b).

4.a.2 Validity

KSs operating on the most valid data should be executed first. This might also be called the "garbage in, garbage out" principle since the validity of the results of a KSI is obviously limited by the quality of its data.

4.a.3 Significance

The inferences made by some KSs are defined, a priori, to be more significant than others, and they are given priority during processing. In Hearsay-II, there was a general tendency to favor the activities of KSs "higher" on the blackboard—the syntactic and semantic interpretation KSs—over the phonological or morphological KSs in any given time segment. This ensured that, for any given segment, Hearsay-II moved to the word-sequence and phrasal levels as soon as possible, instead of expending more resources on additional, alternative low-level analyses of the same data.

4.a.4 Goal satisfaction

KSIs that satisfy goals are preferred to those that do not. Goals are used to dynamically control processing; for example, Hearsay-II can establish goals of generating new word hypotheses for time segments of the utterance, or rejecting a top-down hypothesis by comparing it with data. Hearsay-II had a KS specifically to establish these goals. Note that, when reasoning under uncertainty, the word "goal" has a slightly different meaning than in planning and problem solving systems such as STRIPS (Fikes, Hart, and Nilsson, 1972). In uncertain problems, we cannot be sure that our actions will achieve our goals, nor can we be certain that, after taking actions, our goals have been achieved. For example, Hearsay-II might perform actions that it *expected* to find the correct interpretation for a word in a time segment, but it had no guarantee that the action was actually the best one for the task, nor could it know whether the hypothesized word was actually the correct one. In planning and problem solving systems, in contrast, uncertainty is ignored and goals are merely states that are achieved (or not) by sequences of actions with completely predictable effects.

4.a.5 Competition

Given the choice among several alternative actions, choose the "best" one. All KSIs compete in the sense that some will be judged worthwhile and some will not; but some compete in the stronger sense that they offer the same result. If one executes, it makes the others superfluous. It is very important to recognize this situation since, as we noted above, the efficiency of problem solving depends on avoiding actions that have no *potential* utility. It is usually impossible to predict the exact outcome of an action, but efficiency can be enhanced if one knows enough about potential outcomes to believe that an action is likely to be superfluous.

In addition to these criteria, Hearsay-II was able to recognize—and correct—particular situations that often arise in reasoning under uncertainty. These, too, are quite general and warrant discussion.

First, systems that reason under uncertainty need *stopping criteria*, otherwise they can waste resources vainly trying to increase their certainty that a goal has been achieved. (This cannot happen in orthodox planning systems, since there is never any uncertainty that a goal has been achieved.) Hearsay-II was able to recognize *stagnating* positions—places where it could not increase its certainty in hypotheses irrespective of the resources devoted to the task—and redirect its attention to more productive areas. The situation is analogous to a lovestruck teenager sending flowers to a beau who doesn't respond: if one bunch of flowers isn't enough to achieve the goal, maybe two or three or ... At some point, the system must recognize that its actions are not having the desired effect, or that

there is no increase in certainty that the effect has (or has not) been achieved, and it must give up that particular line of attack.[6]

Another common situation involves managing the uncertainty introduced by using categorical reasoning to model reasoning under uncertainty. In Hearsay-II, as in many other systems, thresholds are established to treat inherently continuous quantities as discrete subranges; for example, a KSI might be allowed to execute only if its overall desirability rating is greater than, say, 75. Thresholds like these introduce uncertainty in several ways. First, there is uncertainty inherent in the data from which the KSI's rating is derived, so it is an uncertain number. Second, the threshold, 75, is set empirically and might be wrong. Perhaps 65 would give better performance. Third, the threshold may be appropriate for most situations that arise, but is overly strict in some important cases. Hearsay-II managed all these sources of uncertainty by dynamically adjusting thresholds. A particularly important tactic is to relax thresholds when the system becomes quiescent, that is, when it has not drawn any conclusions—and cannot draw any further conclusions—that satisfy the threshold.

These tactics—focusing on desirable KSIs, and avoiding stagnation and quiescence—are essential for managing uncertainty, not only in noisy signal-interpretation tasks such as speech understanding, but in many other tasks as well. The blackboard architecture developed for Hearsay-II had impressively flexible control:

> ... [These] principles and mechanisms ... provide a parameterized framework for the elaboration of numerous alternative "macroscopic" policies for attentional control in the speech understanding problem. Each of the typical sorts of heuristic problem solving policies can be realized by simple policy modules that manipulate goal utilities and thresholds and respond to quiescence and stagnation in policy-specific ways. (Hayes-Roth and Lesser, 1977, p. 27).

For example, a global view of Hearsay-II's behavior has been called *island-driving*, because the system first establishes hypotheses that are fairly certain (islands of certainty), and then tries to "fill in" the uncertain areas between them.[7] This is an ideal strategy for interpreting noisy data, particularly if the knowledge that generates the interpretation is itself uncertain. But it only works if the interpretations have rich, internal structure and the problem solver can iteratively build the complete interpretation from partial interpretations, guided by constraints on the structure. Blackboard systems represent structure at many different levels: words have internal structure determined by the morphophonemic rules of language, phrases have structure determined by the syntax of the language, and so on. Hearsay-II was able to exploit these and other kinds of internal structure to extend islands of certainty into areas of the utterance that were less certain.

However, while it was possible to specify a variety of control strategies for Hearsay-II, it was not easy. All control was mediated by a complex equation for computing the worth of KSIs. This equation included parameters that represent desirability criteria, empirically-adjusted tuning parameters, and factors that affect the degree to which processing is depth-first or breadth-first, and that keep the system from stagnating or becoming quiescent. Unfortunately, control strategies are difficult to "read" from the settings of numerous parameters. It is hard to see the dependencies between parameters, so the effects of modifying them cannot always be predicted, nor is it clear how to change parameter settings to achieve desired results:

> A significant amount of tuning of the focusing parameters has been attempted. Nevertheless, the current parameter values are probably not optimal, and it seems clearly impossible to determine what optimal values are. ... Only the general focusing problem and our suggested general approaches appear universally valid; statements regarding the validity of particular parameter settings must await major breakthroughs in the development of our mathematical models and analytical techniques. (Hayes-Roth and Lesser, 1977, p. 34).

But although numeric parameters in Hearsay-II are combined in a single, complex equation, this *implementation* of the system is not critical to blackboard *architectures* in general or the Hearsay-II

[6] Other tactics are required for situations where there is no indication at all, probabilistic or otherwise, that the actions are having the desired effect.

[7] It is interesting to note that this apparently global strategy "emerges" from a sequence of local control decisions.

approach in particular. The blackboard architecture embodies powerful ideas about managing uncertainty by sophisticated control *irrespective of its implementation.*

More recently, researchers have developed blackboard systems with explicit control. An elegant approach involves a division between domain blackboards and control blackboards. For example, the BB1 system (Hayes-Roth, Garvey, Johnson, and Hewett, 1986) has a control blackboard on which plans are formulated. As the plans are executed they result in changes to both the domain blackboard and the control blackboard. The plans are explicit and declarative, easily read and easily explained.

5 Sophisticated control in rule-based systems: Metarules and NEOMYCIN

Contemporary with developments in blackboard systems were some efforts to improve the control of rule-based systems. Implicit, fixed strategies such as breadth-first backward chaining often led to bizarre behaviors. For example, MYCIN asked questions in strange orders that made little sense to physicians. It appeared to be "jumping around" instead of following the kind of diagnostic strategy used by physicians. Clancey puts it this way:

> At a certain level, MYCIN's reasoning is arbitrary, lacking the focus on hypotheses we find in people. ... MYCIN does not focus on a particular hypothesis as it goes through its (implicit) tree of diseases. When it considers types of meningitis or organisms, the types are considered arbitrarily, based on the order in which rules are entered into the program. The program proceeds systematically from infection to meningitis to bacterial meningitis to organism, but the process is unordered at each level of refinement with regard to children. (Clancey, 1986, p. 43).

Any arbitrary ordering is necessarily inefficient: The system will necessarily ask questions and perform tests that could have been avoided if it focused attention and controlled its inferences more carefully. Consider the following example:[8] A physician can perform two tests, A and B. A is diagnostic (will confirm or disconfirm) with respect to disease X, but says nothing about diseases Y or Z. B is diagnostic with respect to disease Y but says nothing about disease X or Z. A costs \$10 and B costs \$100. The cost of diagnosis thus ranges from \$10 to \$110 if A is performed before B, and between \$100 and \$110 if B is performed before A. If the patient has disease X, then an arbitrary ordering of the tests can waste \$100. Interestingly, if the patient has disease Y or Z then the cost of the "good" strategy—doing A before B—is \$10 higher than the cost of the "bad" one. One would not call this \$10 wasted, but rather the price paid to potentially save \$100.

Clancey came upon the problem of implicit, arbitrary control because he wanted to teach students medical reasoning but found it impossible to justify MYCIN's diagnostic strategy—primarily because exhaustive backward chaining with arbitrary rule ordering is not a diagnostic strategy. My concern for the issue has a different source: in medicine and other domains, the cost of inefficiency is measured not only in terms of machine cycles but, more importantly, in terms of money, time, and sometimes human lives. Nonetheless, the mechanisms developed by Clancey and others to make rule-based systems follow computationally efficient and meaningful lines of reasoning apply also to the task of minimizing the costs to humans of bad strategies.

One attempt to get these kinds of control strategies from backward chaining systems is Randy Davis' idea of metarules (Davis and Buchanan, 1984). These are production rules, just like the inferential knowledge in MYCIN, but they are used to control inferences by ordering and pruning potential rule firings. Two are illustrated here:

IF the patient's age is over 60 and
there are rules that mention high risk and
there are rules that mention low risk
THEN it is very likely (.8) that the former should be
used before the latter

[8] Due to Glenn Shafer, personal communication, April 10, 1987.

IF the patient has a high fever and
there are rules that mention viruses
THEN it is likely (.7) that these rules will not be useful

The first illustrates how metarules can order rule-firings: the system will first execute rules that deal with high-risk hypotheses. The second shows how potential rules can be pruned: since viral infections rarely cause high fevers, the rules that mention viral infections can be ignored.[9]

Metarules were the first step towards sophisticated, explicit control for backward chaining systems. Another advance is found in Clancey's NEOMYCIN system (Clancey and Letsinger, 1984). Whereas MYCIN's backward chaining algorithm was implicit in its interpreter, in NEOMYCIN every rule is fired for an explicit reason. The reasons are to be found in the way medical knowledge is *organized.* For example, if medical knowledge is organized to make explicit the *subclass* relationship that holds between, say, heart disease and enlargement of the heart, then that relationship can be the basis of a strategy to refine general hypotheses by seeking evidence for their subclasses.

Instead of a single, implicit, uniform control structure, NEOMYCIN used a set of procedures called *diagnostic tasks.* Roughly 20 diagnostic tasks made up an entire diagnostic procedure. The tasks are organized hierarchically; for example, the *pursue hypothesis* task is executed by running two other tasks, *test hypothesis* and *refine node.* The latter task exploits the subclass relation as described above. Diagnostic tasks are little programs coded as meta-rules for accomplishing aspects of diagnosis.

In sum, NEOMYCIN took several important steps away from MYCIN's view of control: It replaced the implicit backward chaining interpreter with explicit diagnostic tasks, many of which capitalize on relationships between objects in the medical domain—classes of disease, specific diseases, pathophysiological states, symptoms—to guide diagnosis.

6 Controlling questions: MUM and MU

This section describes the evolution of a system developed in the author's laboratory for managing uncertainty by control. Specifically, the work addresses the problem of managing uncertainty by efficient control of evidence-generating actions. This kind of reasoning is found in physicians' diagnostic workups, and is illustrated here with a workup for angina. The architecture of the MUM system, which was developed for this and related diagnostic tasks, is then discussed. MUM made explicit a class of control features: those that deal with degree of belief. The expert and knowledge engineer could easily define control features that represent the current and potential future degrees of belief in hypotheses, as well as control features that represent current and potential differences in degrees of belief between pairs of hypotheses. Other control features were not so easily defined, so control in MUM was limited. The MU system, which followed MUM, rectifies this.

6.a MUM—Managing Uncertainty in Medicine

We have said that the efficiency of reasoning under uncertainty depends on three components of control: focus of attention, control of inferences, and control of questions. Efficiency is a heuristic criterion that says, roughly, one should not pay too much for one's certainty. The costs of certainty can be measured in many ways, so efficiency will have different interpretations in different tasks. For example, in real-time control problems (e.g., air-traffic control) the main determinants of efficiency may be focus of attention and control of inferences: if the system cannot control its inferences—if it "thinks too much"—then it will miss important real-time deadlines.

But in medical diagnosis, control of inferences is less important to efficiency than control of actions. A physician manages uncertainty by asking questions, performing tests, and taking other actions that have costs measured in terms of time, money, and discomfort. These costs are far more serious than the cost of wasted machine cycles; thus, control of actions is more important than control of inferences.

[9] The author confesses that the latter rule is of his own making and may not be medically accurate.

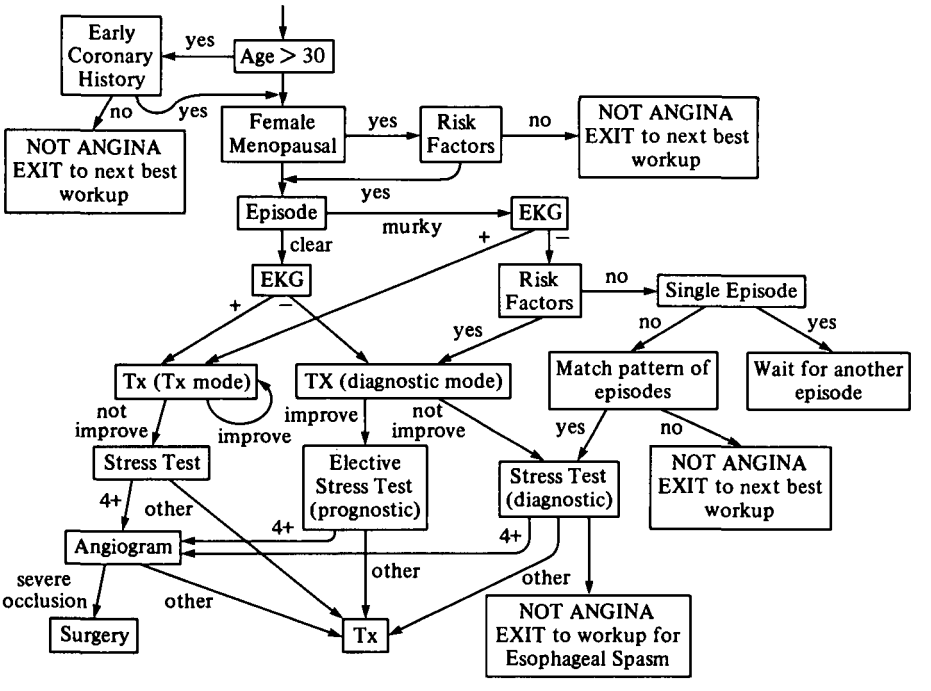

Figure 2 Workup for Angina. NOTE - Tx refers to non-surgical treatment

A *workup* is a sequence of questions, physical palpations, auscultations, tests, and treatments by which physicians make diagnoses. MUM is a program for generating workups, that is, for planning the actions that lead to diagnoses. A workup is a conditional sequence of actions because the outcomes of the actions cannot be known in advance; they depend on the patient's disease. Figure 2 illustrates a workup for angina that was acquired by debriefing an expert internist for several days. MUM does not have workups like Figure 2 stored in any internal data structures; rather, its goal is to generate a sequence of questions, tests, and treatments in a way that conforms to an expert's workup. The emphasis in MUM is on getting the workup right, that is, asking only the relevant questions in the right order.

The planning that underlies a workup graph is illustrated in the context of one path through the expert workup graph. The physician asks a few questions about age, sex, and the episode of chest pain, then orders an EKG. If the EKG does not confirm angina, then the physician will prescribe therapy. If the patient's symptoms abate, an elective stress test is suggested. If the stress test suggests severe coronary artery disease an angiogram is done, possibly followed by surgery. This path contains many examples of managing uncertainty; one notable aspect is that a completely diagnostic test—the angiogram—is delayed as long as possible. Clearly, diagnostic certainty is just one of several goals that remain balanced throughout the workup.

To produce the sequence of questions highlighted in Figure 2, an expert system must reason extensively about its uncertainty and the costs and utility of evidence: The questions about age and sex cost nothing and yet potentially rule out angina. Similarly, questions about the episode of pain can rule out angina (though this is not shown in Fig. 2) and are free. The EKG can potentially confirm angina, although this is unlikely since it detects angina only if the patient is having an attack. Nonetheless, the EKG can provide other evidence (not shown in Fig. 2) pertinent to the angina hypothesis, and is very inexpensive. Thus, the first few questions are asked because they are cheap and potentially disconfirm or confirm a dangerous disease. They are the most efficient actions during the early stages of diagnosis.

If the EKG is negative, the next step is to treat the patient, even though the diagnosis is not certain. Treatment satisfies several goals: it keeps the monetary cost of diagnosis low, minimizes the risk of diagnosis, protects the patient in the event his disease worsens, and provides evidence for or against the disease hypothesis. Abatement of symptoms will provide strong but not confirming evi-

dence for angina. Despite this residual uncertainty the physician may continue treatment for months or more until the patient's condition worsens, since the treatment is inexpensive and has no serious side effects.

Treatment maintains a balance between the cost of evidence and its potential utility. Before treatment, the physician has enough evidence to give it a try. After observing relief of symptoms (and inferring the treatment to be the cause), the physician has enough evidence to continue treatment indefinitely. Treatment is the most efficient action at this point in the workup: other actions might produce stronger evidence (e.g., a stress test), but that evidence is actually stronger than necessary. If, instead of treating the patient, the physician ordered a stress test, the likelihood is that the next step would be treatment. The physician only needs enough evidence to warrant treatment, and this evidence is potentially supplied by treatment itself.

The principle of efficiency underlies all such workups. Actions on a path through a workup graph must be efficient in the sense of providing adequate evidence at a given cost (in terms of money, health, quality of life, etc.) given what is already believed about a patient. This latter criterion explains why, for example, the workup delays an angiogram: a physician does not know enough about the patient's disease to perform an angiogram until the disease is practically confirmed; although the tradeoff between the cost and diagnosticity of the test, given this degree of belief, is acceptable.

6.b MUM architecture

The MUM system generates questions, tests, and treatments in a sequence that conforms to a path through an expert workup graph (Cohen, Day, DeLisio, Greenberg, Kjeldsen, Suthers, and Berman, 1987). Several aspects of its architecture facilitate efficient management of uncertainty through control of questions.

Viewed structurally, MUM's knowledge base is a large inference net with disease-nodes at the top and data-nodes at the bottom. Intermediate *clusters*, which are clinically-significant groupings of evidence, reside in between (see Fig. 3). Data support or detract clusters, which support or detract disease hypotheses. These objects are linked by *roles of evidence*, such as *potentially-confirming* and *potentially-detracting*, that enable MUM to reason about the utility of a cluster in supporting (or detracting) the cluster or disease hypothesis above it in the inference net.

Viewed functionally, MUM's task is to cycle through a loop that first establishes a focus of attention; then decides which question, test, or treatment to invoke, given the current state of the inference network; then propagates the result of this action through the inference network, updating the parameters on which control decisions depend in the next cycle.

A closeup of part of the inference net shows that the data, clusters, and disease hypotheses have internal structure (see Fig. 3). An important component of these nodes is a local *combining function* for evidence. The level of belief in a cluster or disease hypothesis depends on the levels of belief in the clusters that support or detract from it. Local combining functions make these dependencies explicit. For example, if there is some reason to believe the patient is at risk for angina, then some support is provided for the angina hypothesis. If the EKG shows ischemic changes during an episode of substernal pain, then angina is confirmed. On the other hand, if the postprandial cluster is supported, then angina is detracted. The postprandial cluster is confirmed if substernal pain occurs after eating, and it is disconfirmed if the pain is not brought on by eating.

Two aspects of the inference network support management of uncertainty by control of actions. First, MUM recognizes just 7 levels of belief: confirmed, strongly-supported, supported, unknown, detracted, strongly detracted, and disconfirmed. These discrete values make it easy to write local combining functions. Second, local combining functions support reasoning about the effects of gaining evidence. For example, MUM can reason that the postprandial cluster is potentially-supporting for esophageal spasm and potentially-detracting for angina. Discriminating clusters like these could be a focal point in a diagnosis because they are efficient—you get two pieces of evidence from one question. Alternatively, if the esophageal spasm hypothesis were already strongly-

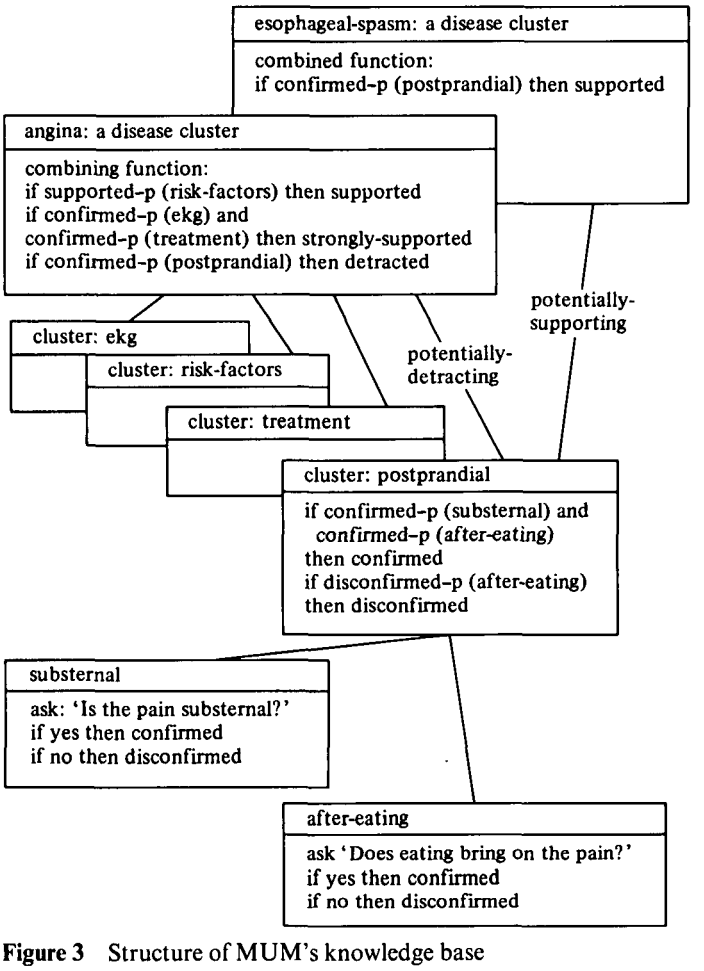

Figure 3 Structure of MUM's knowledge base

supported by some other evidence, MUM would reason that searching for evidence to confirm the postprandial cluster has no marginal utility.

MUM depended on several parameters in addition to level-of-belief to select focus of attention and actions for a workup. These included the dangerousness of hypotheses and costs of tests. But although simple, declarative representation were developed for local combining functions for levels of belief, equal effort was not devoted to other control parameters, so control was difficult to specify and modify. This experience led to an empty version of MUM, called MU, that makes it easy to specify declaratively any control parameters required to efficiently manage uncertainty.

6.c The MU architecture

Structurally, MU further generalizes the inference net representation of MUM. Many control parameters can be propagated by local combining functions, just as level-of-belief was propagated with MUM's network. Functionally, MU provides the expert and knowledge engineer with a language to represent the *state* of a problem, and a set of functions to query aspects of that state. With these abilities, it has been easy to reimplement and then modify MUM's original control structure (Cohen, Greenberg and DeLisio, 1987).

MU makes it easy to define new control parameters in terms of others; for example, the parameter *critical* is defined as *at-least-supported and dangerous*. The new parameter is then associated with an object in MU's network, say *disease*, and its definition is inherited by all diseases. The *critical* parameter is composed of a static parameter and a dynamic one: the value of at-least-supported changes with level of belief, but the value of dangerous is set a priori.

MU has mechanisms for answering five classes of questions about the values of control parameters. The classes are

6.c.1 Questions about current state

For example, what is the monetary cost of an EKG? is esophageal-spasm triggered? Is angina currently critical?

6.c.2 Questions about how to change features

For example, how can I increase the level of belief in angina? How can I decrease belief in gallbladder at low cost?

6.c.3 Focusing questions

For example, what is the set of all diseases that are both triggered and dangerous? Which clusters support angina and are themselves at least supported?

6.c.4 Effect questions

For example, for which diseases does age affect level of belief? Does sex affect level of belief in angina?

6.c.5 Questions about multiple feature changes

For example, what discriminates between angina and esophageal-spasm?

Although these examples of queries are from the medical domain, the MU architecture is domain independent. We believe that any system that manages uncertainty will need to answer at least these five classes of questions about the state of a problem.

MU requires the knowledge engineer to specify control strategies, but it facilitates this by providing easy access to the definitions and values of control parameters and the functions that revise their values. MU does not predispose the knowledge engineer to any particular control strategy; by design, knowledge engineers are encouraged to seek out and implement in MU the control strategies specific to task domains. Others, notably Clancey (1986), Hayes-Roth et al. (1986), and Bylander and Mittal (1986) have based control on a wide variety of declarative control parameters but have taken the additional step of specifying some aspects of control structures for broad classes of tasks.

7 Conclusion

Uncertainty requires us to consider how we solve problems. From this perspective, reasoning under uncertainty is a *control* problem, a task for the component of expert systems that controls focus of attention, inferences, and actions. Although this perspective does not directly address the conventional issues in reasoning under uncertainty (e.g., how to calculate degrees of belief, assumptions of the calculi, etc.) it does give us a way to think about how a problem solver should behave when it is uncertain. It also leads to the view that strategies for managing uncertainty by control are part of the expertise of domains. It suggests that the way to build expert systems to reason about uncertain problems is to acquire problem solving strategies from experts.

Unfortunately, experts do not speak in terms of focus of attention, control of inferences, and control of actions. Their problem solving strategies must be inferred from their behavior and implemented in these terms. This task is arduous enough that, until recently, most expert systems had implicit, inflexible control strategies. Managing uncertainty by control requires the invention of

architectures that support explicit, flexible control. The designers of Hearsay-II took an important step in this direction with the blackboard architecture, which offered extremely flexible, if implicit, control. Explicit control of limited flexibility was achieved by Davis' meta-rules, which could implement some control strategies by ordering and pruning the activations of domain rules.

One way to reduce the gap between expert problem solving strategies and their implementation in control strategies is to define explicit, task-level control features. In Hearsay-II, control features scored knowledge source instantiations and scheduled their execution. They included the quality of evidence, the a priori importance of a knowledge source, and measures of quiescence and stagnation. In Davis's work, domain-specific control features defined the conditions under which meta-rules could order and prune rules, and included terms such as "high-risk" and "age", and assessments of the outcomes of the domain rules such as "concludes the infection type is viral".

More recently, researchers have identified generic tasks that are more general than, say, diagnosing bacteremia infections but more specific than weak methods such as generate-and-test. Tasks such as diagnosis, design, configuration-checking, and prospective reasoning have all been proposed as the basis of task-level architectures. Whereas MYCIN was a domain specific program for diagnosing bacteremia infections, NEOMYCIN and MU are designed as architectures for diagnostic tasks in general. The link between this trend and managing uncertainty by control is that tasks are distinguished by their strategies: different tasks have characteristically different ways of managing uncertainty.

This implies that task-level architectures for diagnosis will have a set of control features that are specific to diagnosis (but not to any particular diagnostic domain) and that may be very different from those that characterize design decisions. For example, NEOMYCIN draws control features from the hierarchical organization of knowledge that underlies many diagnostic tasks. Pathologies have a natural hierarchical organization; for example, heart disease is a kind of cardiovascular problem, and can be further subdivided into angina, myocardial infarction, arythmia, and so on. This structure is exploited by a control strategy that seeks evidence for one or more classes of pathologies and then differentiates them. The control strategy, and the hierarchical organization of knowledge on which it depends, are not specific to a particular diagnostic task but are general to all diagnosis.

Control features are task-specific in MU, too. They include characteristics of the current state of knowledge, such as the degree of belief of a hypothesis, or whether a hypothesis is triggered; characteristics of actions, such as their costs and potential outcomes; conjunctive predicates that apply to hypotheses, such as "hypotheses that are dangerous and supported"; and conjunctive predicates that apply to actions, such as "inexpensive and potentially-confirming". Control features in NEOMYCIN and MU are by design intended to control diagnostic reasoning, without committing to a particular domain of diagnosis.

One argument for task-specific strategies for managing uncertainty by control is that tasks require different interpretations of the criterion of efficiency. Efficiency requires strategies to maximize certainty for given costs and minimize costs for given levels of certainty, but the interpretation of "cost" depends on one's task. In particular, some tasks measure costs in terms of computing time while others emphasize the costs of actions. The former are appropriate in domains where the space of potential inferences increases combinatorially (e.g., the space of hypotheses in Hearsay-II) and in real-time situations where "thinking too much" results in missed deadlines. Strategies of the latter type are appropriate in tasks such as diagnosis; for example, a medical diagnosis system ought to consider the ramifications of an action before committing to it even though the space of ramifications is combinatorially-increasing. In many situations, however, both kinds of costs will be relevant, and expert problem-solving strategies will specify all three components of control strategies: control of inferences, control of actions, and focus of attention.

In sum, our ability to reason in uncertain domains depends on our ability to acquire, represent, organize, maintain, and modify our implementations of expert problem-solving strategies. Traditional approaches to uncertainty, such as probability theory and decision theory, will undoubtedly play a role in this emerging technology. But the greatest impact will come from providing knowledge engineers explicit representations and interpreters for problem-solving strategies, allowing

them to ask and implement the answers to the following question: You're uncertain, now what are you going to *do* about it?

8 Structured bibliography

8.a Combining evidence

Much work in AI on reasoning under uncertainty is concerned with combining evidence. This paper instead views reasoning under uncertainty as a control problem. The following references provide some coverage of techniques and issues in combining evidence. Shortliffe and Buchanan (1984) discuss MYCIN's combining methods—often criticized as ad hoc, but a basic reference nonetheless. Shafer's book and papers (1976, 1984) describe the Dempster-Shafer approach, one of the major paradigms in the field. Duda, Hart, and Nilsson (1974) present one of the first applications of Bayesian inference in the Prospector system, and Pearl's paper (1982) and Quinlan's (1983) raise problems and solutions for Bayesian methods. Lowrance and Garvey (1983), and Gordon and Shortliffe (1984) offer illustrative examples of reasoning with the Dempster-Shafer method. Because the field has expanded so rapidly, one feels compelled to include review articles, for additional coverage, in a bibliography like this. Szolovits and Pauker's (1978) article is a classic comparison of probabilistic and nonprobabilistic reasoning in medical AI systems. Bonissone (1985) offers some criteria by which to judge methods for reasoning under uncertainty. Cohen and Gruber (1985) is a survey of techniques. Henrion raises the question of whether probability has a role in expert systems (1986). Finally, we include some references on nonstandard approaches to uncertainty. Cohen's theory of endorsements argues for symbolic representations of reasons to believe and disbelieve propositions; it is illustrated in the paper by Sullivan and Cohen (1985). Cohen, Shafer and Shenoy (1987) suggest using global numeric combining functions as approximations, augmented by MUM-style local combining functions when dependencies among data are discovered. Shafer and Tversky (1986) offer a provocative new look at the role of probability in expert systems; they propose that the probability of events is judged by comparison with canonical examples which provide the semantics or interpretation of the probabilities.

[1] Bonisonne, P. (1985) Reasoning with uncertainty in expert systems. *International Journal of Man-Machine Studies*, 22:3.

[2] Cohen, P. (1985) *Heuristic reasoning about uncertainty: an artificial intelligence approach.* London: Pitman.

[3] Cohen, P. and Gruber, T. (1985) Reasoning about uncertainty: A knowledge representation perspective. In John Fox (Ed.) *Pergamon Infotech State of the Art Report.*

[4] Cohen, P., Shafer, G. and Shenoy, P. (1987) Modifiable combining functions. *COINS Technical Report 87–43*, Department of Computer and Information Science, University of Massachusetts, Amherst, MA.

[5] Duda, R.O., Hart, P.E and Nilsson, N. (1976) Subjective Bayesian methods for rule-based inference systems. *Technical note 124*, AI Center, SRI International, Menlo Park, CA.

[6] Gordon, J. and Shortliffe, E.H. (1984) The Dempster-Shafer theory of evidence and its relevance to expert systems. *The MYCIN Experiments of the Stanford Heuristic Programming Project*, Reading, MA: Addison-Wesley.

[7] Henrion, Max (1986) Should we use probability in uncertain inference systems. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. pp. 320–331.

[8] Lowrance, John D. and Garvey, Thomas D. (1983) Evidential reasoning: an implementation for multisensor integration. *Technical Note 307*, AI Center, SRI International, Menlo Park, CA.

[9] Pearl, J., Leal, A. and Saleh, J. (1982) GODDESS: A Goal-Directed Decision Structuring System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, pp. 250–262.

[10] Quinlan, J.R. (1983) INFERNO: A cautious approach to uncertain inference. *Computer Journal*, vol. 26.

[11] Shafer, G. (1976) *A mathematical theory of evidence.* Princeton: Princeton University Press.

[12] Shafer, G. (1984) Probability judgment in artificial intelligence and expert systems. *University of Kansas Working Paper 165*, Lawrence School of Business, Lawrence, Kansas.

[13] Shafer, G. and Tversky, A., (1986) Languages and designs for probability judgment. *Cognitive Science.* in press.

[14] Shortliffe, E.H. and Buchanan, B.G. (1984) A model of inexact reasoning in medicine, Rule Based Expert

Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, Reading, MA: Addison-Wesley.

[15] Sullivan M. and Cohen, P.R. (1985) An endorsement-based plan recognition program. *Proceedings of the International Joint Conference on Artificial Intelligence*. pp. 475–479.

[16] Szolovits, P. and Pauker, S.G. (1978) Categorical and Probabilistic Reasoning in Medical Diagnosis. *Artificial Intelligence*, vol. 11, pp. 115–144.

8.b Generic tasks and task-level architectures

A recent trend in expert systems is to identify generic tasks that are more general than domain-specific problems but more specific than weak methods. Diagnosis is one such task, design and configuration are others. Once generic tasks are identified, task-level architectures can be constructed to facilitate building expert systems.

The idea of a task-level architecture is defined in Gruber and Cohen (1987b) Clancey and Chandrasekaran each identified diagnosis as a generic task and described its characteristics in domain-independent terms. Clancey (1985) casts diagnosis as heuristic classification; Chandrasekaran (1986) is a discussion of several generic tasks. Bylander and Mittal (1986) describe a task-level architecture for diagnostic reasoning called CSRL. Clancey (1986) documents the development of the Heracles task-level architecture from MYCIN, through GUIDON and NEOMYCIN. Marcus, McDermott, and Wang (1985) is an excellent example of the separation between task-level and implementation-level constructs: They discuss a knowledge-acquisition tool for a task-level architecture for configuration tasks, and show how this knowledge is automatically compiled into its OPS5 implementation. A similar illustration, for a different task, is given in Kahn, Nowlan, and McDermott (1985). Hayes-Roth, Garvey, Johnson, and Hewitt describe a hierarchy of task-level architectures based on the blackboard. Gruber and Cohen (1987a) discuss principles of design for acquisition, the idea that architectures should be designed from the outset to make it easy to acquire knowledge from experts. These principles are illustrated in the context of a hierarchy of tools for the MUM/MU systems in Gruber and Cohen (1987b). Cohen, Greenberg, and DeLisio (1987) describe MU as a task-level architecture for prospective reasoning.

[1] Chandrasakeran, B. (1986) Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design. *IEEE Expert*, Fall, pp. 23–30.

[2] Bylander, T. and Mittal, S. CSRL: A language for classificatory problem solving and uncertainty handling. *AI Magazine*. 7(3): 66–77, 1986.

[3] Clancey, W.J. Heuristic Classification. *Artificial Intelligence*. 27: 289–350, 1985.

[4] Clancey, W.J. From GUIDON to NEOMYCIN and HERACLES in twenty short lessons. *AI Magazine*. 7(3): 40–60, 1986.

[5] Cohen, P.R., Greenberg, M., DeLisio, J. (1987) MU: A development environment for prospective reasoning systems. *AAAI–87*, July, 1987, Seattle, WA.

[6] Gruber, T. and Cohen, P.R. (1987a) Design for acquisition: principles of knowledge system design to facilitate knowledge acquisition. *The International Journal of Man-Machine Studies*. forthcoming, Spring, 1987.

[7] Gruber, T. and Cohen, P.R. (1987b) Knowledge engineering tools at the architecture level. (revised version) *10th International Joint Conference on Artificial Intelligence*. Milan, Italy. August.

[8] Hayes-Roth, B., Garvey, A., Johnson, M.V. and Hewett, M. A layered environment for reasoning about action. Report No. KSL 86–38. Computer Science Department, Stanford University.

[9] Kahn, G., Nowlan, S. and McDermott, J. MORE: An intelligent knowledge acquisition tool. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August, 581–584.

[10] Marcus, S., McDermott, J. and Wang, T. Knowledge acquisition for constructive systems. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August, 1985. pp. 637–640.

8.c Problem-solving, decision analysis, and planning

Viewing reasoning under uncertainty as a control problem suggests the AI literature on problem solving and planning, as well as the decision analysis literature. Problem solving and planning is

concerned with the control of actions, and decision analysis is concerned with the utility of actions relative to the probability of their outcomes. STRIPS (Fikes, Hart, and Nilson, 1972) is one of the earliest planners and serves as a comparison. A classic paper in planning is Sacerdoti (1979); a survey of the planning literature is given in Cohen and Feigenbaum (1982). Stefik (1980) discusses constraint-based planning in the context of his MOLGEN system; the topic is also covered in Cohen and Feigenbaum (1982).

Two basic references on decision analysis are Raiffa (1970) and Howard (1966). The main problem with decision analysis for reasoning about actions is it requires a complete combinatorial model—too much data for any but the simplest situations. Fox (1985) describes extensions to knowledge-based systems to facilitate decision making. A heuristic, constructive model of decision-making, more appropriate to AI systems, is discussed in Howe and Cohen (1986).

[1] Cohen, P.R. and Feigenbaum, E.A. (1982) *The Handbook of Artificial Intelligence*, 3, Reading, MA: Addison-Wesley.
[2] Fikes, R., Hart, P. and Nilsson, N. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4): 251–288. 1972.
[3] Fox, John (1985) Knowledge decision making and uncertainty. *Workshop on Artificial Intelligence and Statistics*, Bell Laboratories.
[4] Howard, R.A. Decision analysis: applied decision theory, D.B. Hertz and J. Melese, editors, *Proceedings of the Fourth International Conference on Operational Research*. 55–71, New York: Wiley. 1966.
[5] Howe, A. and Cohen, P.R. 1986. A typology for constructing decisions. *Proceedings of IEEE Conference on Computers and Communications*. Phoenix, AZ, February, 1987.
[6] Raiffa, H. *Decision Analysis: Introductory Lectures on Choices Under Uncertainty*. Reading, MA: Addison-Wesley. 1970.
[7] Sacerdoti, E. (1979) Problem solving tactics. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pp. 1077–1085.
[8] Waltz, D. Generating semantic descriptions from drawings of scenes with shadows. Reprinted in P. Winston (Ed.) *The psychology of computer vision*. New York: McGraw-Hill. 19–92, 1975.

8.d Sophisticated Control

The literature on sophisticated control is burgeoning. Much of it is associated with blackboard architectures since they offer very flexible control. Nii (1986a,b) are good tutorial articles on blackboard architectures. The classic article on Hearsay-II is Erman, Hayes-Roth, Lesser, and Reddy (1980), but it is a bit short on details about the control of the system. Hayes-Roth and Lesser (1977) is more extensive. Hayes-Roth (1985) discusses sophisticated control in blackboards, and Hayes-Roth, Garvey, Johnson, and Hewitt (1986) relate these ideas to hierarchies of architectures (cited above). Durfee and Lesser (1986) describe how the control of blackboard systems can be enhanced by planning.

The literature on sophisticated control in rule-based systems is less extensive, but one must read Davis and Buchanan's (1984) discussion of metarules and Clancey's (1986) chronology of the evolution of Heracles from MYCIN, GUIDON, and NEOMYCIN (cited above). Clancey and Letsinger (1984) describe the NEOMYCIN system from the perspective of making control knowledge explicit for teaching purposes.

MUM was implemented initially as a blackboard system but its architecture is actually a hybrid. It uses a rule-like inference network with a control-blackboard-like planner. MUM and its successor, MU, are described in Cohen, Day, DeLisio, Greenberg, Kjeldsen, Suthers, and Berman (1987), and Cohen, Greenberg, and DeLisio (1987) cited above.

[1] Clancey, W.J. and Letsinger, R. NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching. In W.J. Clancey and E.H. Shortcliffe (Eds.) *Readings in Medical Artificial Intelligence: The First Decade*. Reading, MA.: Addison-Wesley, 1984.
[2] Cohen, P., David Day, Jefferson DeLisio, Michael Greenberg, Rick Kjeldsen, Daniel Suthers and Paul Berman. Management of Uncertainty in Medicine. *International Journal of Approximate Reasoning*, Spring, 1987.
[3] Davis, R. and Buchanan, B.G., Meta-level knowledge. In B.G. Buchanan and E.H. Shortliffe (Eds.) *Rule-*

based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project, Reading, MA.: Addison-Wesley, 1984.

[4] Durfee, E. and Lesser, V. (1986) Incremental planning to control a blackboard-based problem solver. *Proceedings of the Fifth National Conference on Artificial Intelligence*. pp. 58–64.

[5] Erman, L.D., Hayes-Roth, F., Lesser, V. and Reddy, D.R. The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Survey*, 12: 213–253. 1980.

[6] Hayes-Roth, B. (1985) A blackboard architecture for control. *Artificial Intelligence*, vol. 26, pp. 251–321.

[7] Hayes-Roth, F. and Lesser, V. (1977) Focus of attention in the Hearsay-II speech understanding system. *Proceedings of the Fifth International Joint Conference on Articial Intelligence*. Boston, MA. pp. 27–35.

[8] Nii, H.P. The blackboard model of problem solving. *AI Magazine* 7(2): 38–54. 1986a.

[9] Nii, H.P. Blackboard systems part two: Blackboard application systems. *AI Magazine* 7(3): 82–106. 1986b.