MU: A Development Environment for Prospective Reasoning Systems

Paul R. Cohen and **Michael Greenberg** and **Jefferson DeLisio**
Experimental Knowledge Systems Laboratory
University of Massachusetts
Amherst, Massachusetts 01003[1]

Abstract

We describe a style of problem solving, prospective reasoning, and a development environment, MU, for building prospective reasoning systems. Prospective reasoning is a form of planning in which knowledge of the state of the world and the effects of actions is incomplete. We illustrate one implementation of prospective reasoning in MU with examples from medical diagnosis.

I. Introduction

MU is a development environment for knowledge systems that reason with incomplete knowledge. It has evolved from a program called MUM that planned diagnostic sequences of questions, tests, and treatments for chest and abdominal pain [Cohen *et al.*, 1987]. This task is called prospective diagnosis, because it emphasizes the selection of actions based on their potential outcomes and the current state of the patient. Prospective diagnosis is uncertain because the precise outcomes of actions cannot be predicted, in part because knowledge of the state of the patient is incomplete. Yet we have found that physicians have rich strategic knowledge with which they plan diagnoses in spite of their their uncertainty. MU does not provide a knowledge engineer with any particular strategies, but rather provides an environment in which it is easy to acquire, represent, and experiment with a wide variety of strategies for prospective diagnosis and other *prospective reasoning* tasks.

Three goals underlie our research and motivate the MU system. First, MU is intended to provide knowledge-engineering tools to help acquire expert problem-solving strategies. MU allows us to define explicit *control features*, which are the terms an expert uses to discuss strategies. Control features in medical diagnosis include degrees of belief in disease hypotheses, monetary costs of evidence, the consequences of incorrect conclusions, and "intangibles" such as anxiety and discomfort. Some, like degrees of belief, have values that change dynamically during problem solving. MU helps the knowledge engineer define the functions that compute these dynamic values and keeps the values accessible during problem solving. For example, with MU we can easily define a control feature called *criticality* in terms of two others, say *dangerousness* and *degree of belief*, and acquire a function for dynamically assessing the criticality of a hypothesis as its degree of belief changes.

Second, we want to show that strategies enable a prospective reasoning system to produce solutions that are *efficient* in the sense of minimizing the costs of attaining given levels of certainty. MU has no "built in" problem solving strategies, but we have been able to acquire and implement efficient, expert strategies in MU because we can define explicit control features that represent the various costs of actions, as well as the levels of certainty in the evidence produced by actions.

Third, we want to implement in MU a *task-level architecture* for prospective reasoning [Gruber and Cohen, 1987], an environment for building systems that plan efficient sequences of actions, despite uncertainty about their outcomes. After working in the domains of medicine and plant pathology, we now think that many control features pertain to diagnostic tasks in general. Moreover, diagnosticians in many fields seem to use similar strategies to solve problems efficiently. This view is influenced by the recent trend in AI toward defining *generic tasks* [Chandrasakeran, 1986] such as classification [Clancey, 1985] and the architectures that support their implementation. MU shares the orientation toward explicit control efforts such as BB* [Hayes-Roth, 1985, Hayes-Roth *et al.*, 1986] and Heracles [Clancey, 1986] but emphasizes control features that are appropriate for prospective reasoning.

In sum, MU is a tool for representing and providing access to the knowledge that underlies efficient prospective reasoning. This paper begins with an analysis of prospective reasoning, then describes the MU environment first as a program, emphasizing its structure and function, then from the perspective of the knowledge engineer who uses it. As an illustration, we describe how MUM was reimplemented in MU. We conclude with a summary of current work.

II. Prospective Reasoning

Prospective reasoning is reasoning about the question "What shall I do next," given that

1. knowledge about the current state of the world is incomplete,

[1]We thank Carole Beal for many helpful comments on drafts of this paper. This research is funded by DARPA/RADC Contract F30602-85-C-0014 and by NSF Grant IST 8409623.

2. the outcomes of actions are uncertain,
3. there are tradeoffs between the costs of actions with respect to the problem solver's goals and the utility of the evidence they provide,
4. states of knowledge that result from actions can influence the utility of other actions.

An example from medical diagnosis illustrates these characteristics:

> A middle-aged man reports episodes of chest pain that could be either angina or esophageal spasm; the physician orders an EKG, but it provides no evidence about either hypothesis; then he prescribes a trial prescription of vasodilators; the patient has no further episodes of pain, so the physician keeps him on long-acting vasodilators and eventually suggests a modified stress test to gauge the patient's exercise tolerance.

The first and second characteristics of prospective reasoning are clearly seen in this case: Knowledge about the state of the patient is incomplete throughout diagnosis, and the outcomes of actions (the EKG, trial therapy, stress test) are uncertain until they are performed and are sometimes ambiguous afterwards. Less obvious is the third characteristic, the tradeoffs inherent in each action. Statistically, an EKG is not likely to provide useful evidence, but if it does, the evidence will be completely diagnostic. The EKG is given because its minimal costs (e.g., time, money, risk, and anxiety) are offset by the possibility of obtaining diagnostic evidence[2]. Similarly, trial therapy satisfies many goals; it protects the patient, costs little, has few side-effects and, if successful, is good evidence for the angina hypothesis.

The fourth characteristic of prospective reasoning is that states of knowledge that result from actions can affect the utility other actions. This is because the costs and benefits of actions are judged in the context of what is already known about the patient. For example, trial therapy is worthwhile if the EKG does not produce diagnostic evidence, but is redundant otherwise. The outcome of an EKG thus affects the utility of trial therapy. This implies a dependency between the actions, and suggests a strategy: do the EKG first because, if it is positive, then trial therapy will be unnecessary.

Dependencies between actions help the prospective reasoner to *order* actions. We call this planning, though it is not planning in the usual AI sense of the word [Sacerdoti, 1979, Cohen and Feigenbaum, 1982]. The differences are due to the first and second characteristics of prospective reasoning: the state of the world and the effects of actions are both uncertain. The prospective 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 prospective reasoning tend to be short. In contrast, uncertainty is excised from most AI planners 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. All further discussions of planning in this paper refer to the "feel your way" variety, not to "dead reckoning."

Prospective diagnosis requires a planner to select actions based on their costs and utility given the current state of knowledge about the patient. We have described prospective reasoning as planning because the evidence from one action may affect the utility of another. Alternatively, prospective reasoning can be viewed as a series of decisions about actions, each conditioned on the current state of knowledge about the patient. We considered decision analysis [Raiffa, 1970, Howard, 1966] as a mechanism for selecting actions in prospective reasoning, but rejected it for two reasons. First, collapsing control features such as monetary expense, time, and criticality into a single measure of utility negates our goals of explicit control and providing a task-level architecture for prospective reasoning [Cohen, 1985, Gruber and Cohen, 1987]. Second, decision analysis requires too many numbers — 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. But although we do not implement prospective reasoning with decision analysis, MU is designed to provide qualitative versions of several decision-analytic concepts, including the utility of evidence and sensitivity analysis.

III. The MU Environment – An Overview

A coarse view of MU's structure reveals these components:

- a frame-based representation language,
- tools for building inference networks,
- an interface for defining control features and the functions that maintain their values,
- a language for asking questions about the state of a problem and how to change its state.
- a user interface for acquiring data during problem-solving,

With these tools, a knowledge engineer can build a knowledge system with a planner for prospective reasoning. MU does not "come with" any particular planners, but it provides tools for building planners and incorporating expert problem-solving strategies within them.

Among MU's tools is an editor for encoding domain inferences, such as *if EKG shows ischemic changes then angina is confirmed*, in an inference network. MU does not dictate what the nodes in the inference network should represent, except in the weak sense that nodes "lower" in the

[2]This example oversimplifies the reasons for giving an EKG, but not the cost/benefit analysis that underlies the decision.

network — relative to the direction of inference — provide evidence for those "higher" up. However, the nodes in the network are usually differentiated; for example, in Figure 1 some nodes represent raw data, others represent combinations of data (called *clusters*), and a third class represents hypotheses. In the medical domain, data nodes represent individual questions, tests, or treatments. Clusters combine several data; for example, the *risk-factors-for-angina* cluster combines the patient's blood pressure, family history, past medical history, gender, and so on. Hypothesis nodes represent diseases such as *angina*.

Since MU does not provide a planner, the knowledge engineer is required to build one. The planner should answer two questions:

- Which node(s) in the network should be in the focus set, and which of these should be the immediate focus of attention?
- Which actions are applicable, given the focus set, and which of these should be taken?

For example, in the medical domain the focus set might include all disease hypotheses that have some support, and the immediate focus of attention might be the most dangerous one. The potential actions might be the leaf nodes of the tree rooted at the focus of attention (Fig. 1), and the selected action might be the cheapest of the potential actions.

An Inference Net in MU

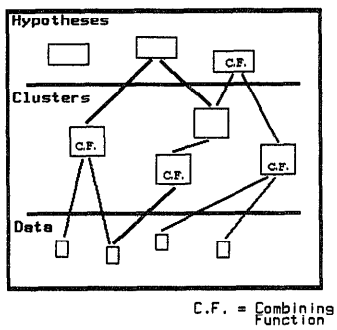

Figure 1: Organization of Knowledge Within Mu

MU provides an interface to help the knowledge engineer define control features such as the degree of belief in hypotheses, the dangerousness of diseases, and the costs of diagnostic actions. It also provides a language with which a planner can query the values of features and ask about actions that would change those values. Planners can ask, for example, "What is the current level of belief in angina?" or "Tell me all the inexpensive ways to increase the level of belief in angina," or even the hypothetical question, "Would the level of belief in angina change if blood pressure was high?"

The relationship between these functions of MU and the functions of a planner are shown in Figure 2. Using MU, a knowledge engineer can: define a control feature such as criticality in terms of other features such as dangerousness and degree of belief; specify a combining function for calculating dynamically the value of criticality from these other features during problem solving; associate criticality and its combining function with a class of nodes, such as diseases, and have each member of the class inherit the definitions; and write a planner that encodes an expert strategy for dealing with critical or potentially-critical diseases. MU facilitates the development of planners, and makes their behavior explicit and efficient, but the *design* of planners, and the acquisition of strategies and the control features on which they depend, is the job of the knowledge engineer.

MU System

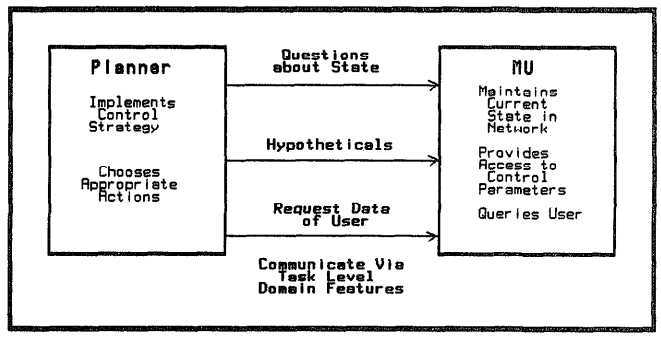

Figure 2: Mu System Schematic

IV. The MU Environment – Features and Combining Functions

Knowledge representation in MU centers around features. Features and their values are the information with which planning decisions are made. Each node in a MU inference network can have several features; for example, the node that represents trial therapy for angina includes features for monetary cost and risk to the patient. Features are defined in the normal course of knowledge engineering to support expert strategies for prospective reasoning. We have identified four classes of features, differentiated by their value types, how they are calculated, and the operations that MU can perform on them:

Static The value of a static feature is specified by the expert and does not change at run time. *Monetary cost* is a typical static feature, as the cost of an action does not change during a session.

Datum The value of a datum feature is acquired at run time by asking the user questions. Data are often the results of actions; for example *EKG shows ischemic changes* is a potential result of performing an EKG.

Dynamic The value of a dynamic feature is computed from the values of other feature values in the network.

The value of each dynamic feature is calculated by a combining function, acquired through knowledge engineering. A dynamic feature of every hypothesis is its *degree of belief* — a function of the degrees of belief of its evidence.

Focus The value of a focus feature is a set of nodes whose features satisfy a user-defined predicate. Focus features are a subclass of dynamic features. In medicine, the *differential* focus feature can be defined as the list of all triggered hypotheses that are not confirmed or disconfirmed.

Feature values can belong to several data types, including integers, sets, normal (one of an unordered set of possible values), ordinal (one of an ordered set of possible values), bpolean, and relational (e.g., isa).

Four operations are defined for features: one can *set* a feature value (e.g., assert that the monetary cost of a test is high) *get* a feature value (e.g., ask for the cost of a test), ask *how to change* a feature value, and ask *what are the effects* of changing a feature value. Planners need answers to these kinds of questions to help them select actions (see Section 5 for further examples.)

All combinations of feature type, value type, and operations are not possible. Figure 3 summarizes the legal combinations.

MU provides an interface for defining features. A full definition includes the feature type, value type, its range of values, and the domain of its combining functions. For instance, the dynamic feature *level of support* is defined to have seven values on an ordinal scale: disconfirmed, strongly-detracted, detracted, unknown, supported, strongly-supported and confirmed. Figure 4 shows the definition of level of support.

Instances of this feature (and others) are associated with individual hypotheses, each of which may have its own, local function for calculating level of support, and its own, dynamic value for the feature[3]. For example, Figure 5 shows part of the frame for the angina hypothesis, encompassing an instance of the level of support feature, and showing a fragment of the function for calculating its value for angina.

Level-Of-Support

Feature-type: Dynamic

Value-Type: Ordinal

Value-restriction: (disconfirmed strongly-detracted
detracted unknown
supported strongly-supported confirmed)

Combination-function-slot: *local to each hypothesis*

Value: *the current level of support of the hypothesis*

Figure 4: Definition of Level-Of-Support

Angina

Feature-list: (level-of-support severity)

Current-level-of-support: strongly-supported

Combination-function:
IF value of ekg is ischemic-changes
THEN angina is confirmed
ELSEIF episode-incited-by contains exertion
AND
risk-factors-for-angina are supported
THEN angina is strongly-supported . . .

Figure 5: Part of the Angina Frame With Local Combining Function

Combining functions calculate values for dynamic features such as level of belief, criticality, elapsed time, and so on. They serve two important functions: First, they keep the state of MU's inference network up-to-date; for example, when the result of an EKG becomes available, the combining function for the angina node updates the value of its level of support feature accordingly.

Second, and perhaps more important from the standpoint of a planner, combining functions provide a *prospective* view of the effects of actions; for example, the combining function for angina can be interpreted prospectively to say that EKG can potentially confirm angina. The same

	Data Types				Questions			
Feature	Number	Set	Ordinal	Normal	Get	Set	How To	Effect Of
static	X	X	X	X	X			
datum	X	X	X	X	X	X		X
dynamic	X		X	X	X		X	
focus		X			X			

Figure 3: Capabilities By Feature Type

[3]Not all feature values are calculated locally, but, for reasons discussed in [Cohen, Shafer, and Shenoy, 1987] and [Cohen *et al.*, 1987] levels of belief are.

point holds for the combining functions for other features: MU can prospectively assess the potential effects of actions on all dynamic features. A planner can ask MU, "If EKG is negative, what changes?" and get back a list of all the features of all data, clusters, and hypotheses that are in some way affected by the value of EKG. The effects of actions are assessed in the context of MU's current state of knowledge (i.e., the state of its network). For example, if an EKG has been given and its results were negative, then MU knows that the answer to the previous question is that nothing changes.

The syntax of combining functions is relatively unimportant provided they are declarative, so MU's question-answering interface can read them, and experts can easily specify and modify them. Currently, combining functions look like rules, but we are experimenting with tabular and graphic forms [Cohen, Shafer, and Shenoy, 1987].

The two major classes of combining functions are *local* and *global*. A local function for a node such as angina refers only to the nodes in the inference network that are directly connected to angina. In contrast, global functions survey the state of MU's entire inference network. Functions for focus features take a global perspective because the value of a focus feature is the subset of nodes in the network whose features satisfy some predicate. For example, Figure 6 illustrates the combining function for the *differential* focus feature. Any node that represents a disease hypothesis, and is triggered, but is neither confirmed nor disconfirmed is a member of the differential.

Differential

feature-list: (focus-feature)

current-focus: (angina prinz-metal ulcer)

combining-function:

Set-of $node$ member-of disease Such-that
$node$ is triggered AND
level-of-support of $node$ is not confirmed AND
level-of-support of $node$ is not disconfirmed

Figure 6: Part of the Global Focus-Feature **Differential**

The knowledge engineer can define many focus features, each corresponding to a class of nodes that a planner may want to monitor. Besides the differential, a planner might maintain the set of critical hypotheses (e.g., all dangerous hypotheses that have moderate support or better), or the set of hypotheses that have relatively high prior probability, or the set of all supported clusters that potentially confirm a particular hypothesis. MU supports set intersection, union, and sorting on the sets of nodes maintained by focus features. A planner's current focus of attention is represented in terms of the results of these operations.

V. MU from the Knowledge Engineer's Perspective

MU is a development environment for prospective reasoning systems. We began our research on prospective reasoning when we were building a system, MUM, for prospective diagnosis [Cohen *et al.*, 1987], and realized that we lacked the knowledge engineering tools to acquire and modify diagnostic strategies. An example will illustrate the knowledge engineering issues in building MU:

MUM had several *strategic phases*, each of which specified how to assess a focus of attention and select an action. One phase, called *initial assessment*, directed MUM to focus on triggered hypotheses one by one and take inexpensive actions that potentially support each. This covered a wide range of situations, and maintained the efficiency of diagnoses by focusing on low-cost evidence, but it made little sense for very dangerous disease hypotheses. For these, diagnosticity — not cost — is the most important criterion for selecting actions. Once the expert explained this, we should have immediately added a new strategic phase, run the system, and iterated if its performance was incorrect. Unfortunately, control features such as criticality and diagnosticity did not have declarative representations in MUM, were implemented in lisp, and could not easily be composed from other control features. Operations such as sorting a list of critical hypotheses by their level of support were also implemented in lisp. Each strategic phase required a day or two to write and debug. From the standpoint of the expert, it was an unacceptable delay.

The MUM project showed us that MU should facilitate acquisition of control features, maintain their values efficiently, and support a broad range of questions about the state of the inference network. MU allows a planner to ask 6 classes of questions:

Questions about **state** are concerned with the current values of features. For example:

Q1: "What is the current level of support for angina?"

Q2: "Is an ulcer dangerous?"

Q3: "What is the cost of performing an angiogram?"

Another class of questions is asked to find out how to achieve a **goal**. Examples of questions about goals are:

Q4: "Given what I know now, which tests might confirm angina?"

Q5: "What are all of the tests that might have some bearing on heart disease?"

These questions help a planner identify relevant actions and select among them. Those that pertain to levels of belief are answered by refering to the appropriate combining functions and current levels of belief. For example, the answer to the question about angina is "EKG," if an EKG has not already been performed (Fig. 5).

Questions about the **effects** of actions allow a planner to understand the ramifications of an action. For example,

Q6: "Which disease hypotheses are affected by performing an EKG?

Q7: "What are the possible results of an angiogram?"

Q8: "Does age have an effect on the criticality of colon cancer?"

MU answers these questions by traversing the relations between actions and nodes "higher" in the inference network. For example, Q6 is answered by finding all the nodes for which EKG provides evidence. The planner may ask either for the *immediate* consequence of knowing EKG, or for the consequences to any desired depth of inference.

Focus questions help a planner establish focus of attention. For example:

Q9: "Give me all diseases that are triggered and dangerous."

Q10: "What are all of the critical diseases for which I have no information?"

Q11: "Are any hypotheses confirmed?"

Questions about **multiple effects** allow the planner to combine the previous question types into more complex queries such as "What tests can discriminate between angina and esophageal spasm?" In this case, the term *discriminate* is defined to mean "simultaneously increase the level of belief in one disease and lower it in an other."

Hypothetical questions allow the planner to identify dependencies among actions. For example, one can ask, "Suppose the response to trial therapy is positive. Now, could a stress test still have any bearing on my belief in angina?"

With the ability to define control features and answer such questions, we quickly reimplemented MUM's strategic phase planner. Most of the effort went into adding declarative definitions of control features and their combining functions to MUM's medical inference network.

VI. Conclusion

MU supports the construction of systems that have the characteristics of prospective reasoning identified in Section 2: Prospective reasoning involves answering the question, "What shall I do next," given uncertainty about the state of the world, the effects of actions, tradeoffs between the costs and benefits of actions, and precondition relations between actions. The six classes of questions, discussed above, help planners to decide on courses of action despite uncertainty. Questions about **state** make uncertainty about hypotheses explicit. **Hypothetical** questions and questions about **effects** make uncertainty about the outcomes of actions explicit. Questions about **goals** and **multiple effects** help a planner identify the tradeoffs between actions. And **hypothetical** questions make dependencies between actions explicit.

We are currently extending MU's abilities in several ways. One project seeks to automate the process of acquiring strategies. It attempts to infer strategies from cases, asking the expert to supply new control features when the current set is insufficient to represent the conditions under which strategies are appropriate. We are also building an interface to help acquire combining functions. This task becomes confusing for the expert and knowledge engineer alike when levels of belief must be specified for combinations of many data. We discuss related work on the design of functions to extrapolate from user-specified combining functions in [Cohen, Shafer, and Shenoy, 1987]. A third project is to implement sensitivity analysis in MU. The goal is to add a seventh class of queries, of the form, "To which data and/or intermediate conclusions is my current level of belief in a hypothesis most sensitive." This will facilitate prospective reasoning by giving the planner a dynamic picture not only of its belief in hypotheses, but also in its *confidence* in these beliefs. With sensitivity analysis the prospective reasoner will be able to find weak spots in its edifice of inferences and shore them up (or let them collapse) before they become the basis of unwarranted conclusions.

References

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

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

[Clancey, 1986] Clancey, W. J. From guidon to neomycin and heracles in twenty short lessons. *AI Magazine*. 7(3):40-60, 1986.

[Cohen *et al.*, 1987] Cohen, P., Day, D., Delisio, J., Greenberg, M., Kjeldsen, R., Suthers, D., & Berman, P. Management of uncertainty in medicine. *International Journal of Approximate Reasoning*. 1(1): Forthcoming.

[Cohen and Feigenbaum, 1982] Cohen, P. R., and Feigenbaum, E. A. *The Handbook of Artificial Intelligence, Vol. 3*. Addison-Wesley, Reading, Massachusetts, 1982.

[Cohen, 1985] Cohen, P. R. Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach. *Pitman Advanced Research Note*. Pitman Publishing, London, 1985.

[Cohen, Shafer, and Shenoy, 1987] Cohen, P. R., Shafer, G., and Shenoy, P. Modifiable combining functions. *EKSL Report 87-05*, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1987.

[Fikes, Hart, and Nilsson, 1972] Fikes, R., Hart, P., and Nilsson, N. Learning and executing generalized robot plans. *Artificial Intelligence* 3(4):251–288, 1972.

[Gruber and Cohen, 1987] Gruber, T. R. & Cohen, P. R. Design for acquisition: principles of knowledge system design to facilitate knowledge acquisition. To appear in the *International Journal of Man Machine Studies*, 1987.

[Hayes-Roth, 1985] Hayes-Roth, B. 1985. A blackboard architecture for control. *Artificial Intelligence*, 26:251–321, 1985.

[Hayes-Roth *et al.*, 1986] Hayes-Roth, B., Garvey, A., Johnson, M.V., and Hewett, M. A layered environment for reasoning about action. *KSL Report No. 86-38*. Department of Computer Science. Stanford University, 1986.

[Howard, 1966] Howard, R.A. Decision Analysis: Applied Decision Theory. In D.B. Hertz and J. Melese, editors *Proceedings of the fourth International Conference on Operational Research*, pages 55–71, Wiley, New York, 1966.

[Raiffa, 1970] Raiffa, H. *Decision Analysis: Introductory Lectures on Choices Under Uncertainty*. Addison-Wesley, Reading, Massachusetts, 1970.

[Sacerdoti, 1979] Sacerdoti, E. Problem solving tactics. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 1077–1085, 1979.