

A Method for Acquiring Strategic Knowledge

COINS Technical Report 88-53

Thomas R. Gruber

May, 1988

Experimental Knowledge Systems Laboratory
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Abstract

This paper presents a method for acquiring strategic knowledge from experts. Strategic knowledge is used by an agent to decide what action to perform next, where actions effect both the agent's beliefs and the state of the external world. It underlies expertise in many tasks, yet it is difficult to acquire from experts and is generally treated as an implementation problem. To acquire strategic knowledge, we have designed an operational representation of strategy, devised a technique for eliciting it from experts, and built an assistant that manages a learning dialog with the expert. The assistant generalizes strategic knowledge with syntactic induction operators, guided by feedback from the expert. We conclude with an analysis of the sources of power and limitations of the approach, and its relation to other research in knowledge acquisition.

To appear in the Proceedings of the Third Knowledge Acquisition for Knowledge-based Systems Workshop, Banff, Canada, November 6-1, 1988.

This research was sponsored by DARPA-RADC contract F30602-85-C0014 and ONR University Research Initiative contract N00014-86-K-1764. Many thanks to Paul Cohen for insightful collaboration and guidance.

1 Introduction

This paper presents a method for acquiring strategic knowledge from experts with machine assistance, in the manner of interactive knowledge acquisition systems such as TEIRESIAS [10], ETS [2], SALT [29], and MOLE [17]. We begin with a description of strategic knowledge, the problem of acquiring it, requirements for a solution, and a brief description of the approach. We trace an example session between an expert and the computer assistant, called ASK, to illustrate the functionality of the approach. We then explain how ASK works, outlining its essential components. We conclude by discussing the sources of power of the method, its scope of applicability, and its relation to existing approaches to knowledge acquisition.¹

1.1 The Problem of Acquiring Strategic Knowledge

Strategic knowledge is knowledge about how to act; more precisely, it is knowledge used by an agent to decide what action to perform next, where actions effect both what is believed by the agent and the state of the external world. A robot applies strategic knowledge to decide between going around or over an obstacle. A lawyer uses strategy in deciding which question to pose in cross examination and which case to cite in defense of an argument. Strategic knowledge underlies expertise in domains where the *process* of problem solving, choosing among possible actions, is critical to performance. It is distinguished from the substantive knowledge of a domain, such as the inferential relationships between data and hypotheses. It is heuristic knowledge that is useful when the precise effects of actions are not known (if the consequences were predictable, then an optimal choice could be found by search). Strategy is also a means for dealing with incomplete knowledge; an agent that can *act* without complete knowledge of the world is more robust.

We have studied the strategic knowledge that physicians bring to bear in the medical work-up: the process of gathering data, assessing the results, and planning treatment. In AI, medical diagnosis is often treated as a classification problem, with substantive knowledge relating findings to diseases. Viewed as a process, however, the medical work-up requires additional knowledge about how to proceed when the data is not all in. Strategic knowledge underlies the physician's expertise at choosing the next question to ask, diagnostic test to run, or therapy to try.²

¹A review of the relationship of this work to the major paradigms of knowledge acquisition research can be found in [20].

²This work has been motivated and informed by experience in building a knowledge system for the prospective diagnosis of chest pain, called MUM [8]. MUM's performance criteria was not to get the right diagnosis (it did), but to do the diagnosis as a physician would, balancing the potential costs of a diagnostic action, such as a diagnostic test or trial therapy, with the evidential and therapeutic benefits.

Although strategy is expertise, strategic knowledge is rarely acquired from experts. The conventional separation of control from inference in knowledge systems treats strategy as an implementation problem, left solely to the knowledge engineer. For domains where strategy plays an important role, it is less productive to “engineer” strategic knowledge than to acquire it. First, if strategic knowledge is implicitly represented by implementation-level primitives, such as numeric priorities, the order of rules or rule clauses, or Lisp functions, then the usual benefits of explicit knowledge representation are lost: explainability, modifiability, and meta-level reasoning. Second, this practice excludes the expert from directly participating in the formulation and debugging of strategic knowledge. Third, the lack of a methodology for acquiring strategic knowledge makes it inefficient to build knowledge systems that depend on strategy, since each system is essentially a custom design.

A primary goal of the research reported here is to devise a method for acquiring strategic knowledge from experts that can be automated, which would facilitate the construction of knowledge systems that act with good reason. We present the method and then discuss the design of the implementation, showing how general principles were applied to this knowledge acquisition problem.

1.2 Objectives and overview of approach

What constitutes a solution to the problem of acquiring strategic knowledge? Initially, we must characterize the behavior that we wish to model and identify the knowledge that is sufficient to produce the desired behavior. We defined strategic knowledge as knowledge used to choose among actions, which emphasizes a view of strategy as iterative control or reactive planning [4,18]. Other views are possible. The implications of the decision to model strategy in this way is discussed in Section 3.

Given a general view of strategic knowledge, we must design or adapt an operational representation that can be executed by a knowledge system to achieve the desired behavior. In practical terms this means designing an architecture for control. We have chosen to represent strategic knowledge with *strategy rules*, inspired by the metarules that represent diagnostic strategy in NEOMYCIN [5]. A knowledge base of strategy rules serves as the control component of a knowledge system. Strategy rules map strategic situations to sets of actions with specified features. At each iteration in a running system, strategy rules that match the current situation choose the actions to perform next.

In addition to an operational representation, we need a technique for eliciting the strategic knowledge from people. A knowledge elicitation technique has two main functions: to ask for the knowledge and to provide a means for the expert to answer. In theory, one could ask the expert to “tell me your strategy” and provide the operational representation as the language in which to reply. In practice, an elicitation technique

must constrain the dialog, so that the system only asks for knowledge it needs, and asks questions in a form that is natural to the expert. Our experience indicates that experts can give *justifications* for their actions [8]. This influences the design of the elicitation medium: it should be a language for describing the rationale for specific strategic decisions. Our system asks the expert why an action was chosen in the context of an actual case, and the expert answers by indicating which factors were relevant to the current choice. The system provides a set of factors to choose from, and if the currently known factors are incomplete, the expert is given an interface for defining new ones, also in the context of justifying actions.

Finally, if we are to *automate* the acquisition of strategic knowledge we need to build an assistant that can structure the dialog with the expert and formulate strategy in the operational representation. The assistant needs to know what knowledge is required, how to elicit it from the expert, and how to put it in operational form. We have developed an assistant called ASK that elicits justifications from the expert and generates strategy rules. It uses simple inductive operators to generalize what it has acquired, and asks the expert to verify the generalizations.

Details about the representation, the elicitation technique, and the learning process are presented in Section 2. First we present a sample knowledge acquisition dialog.

1.3 An Example Dialog Session with ASK

This section presents an annotated transcript of a knowledge acquisition session between the expert and ASK. Annotations in *italics* explain what the expert and program are doing. For economy and clarity it is written as a natural language dialog, with ASK prompting for information. In actuality, the expert usually answers by clicking on items in windows and menus, and can take the initiative in the dialog. A snapshot of the screen is shown in Figure 1.

The setting: a work-up for chest pain

The dialog begins during the execution of a knowledge system for the diagnosis and treatment of chest pain. The system already has the substantive knowledge of the domain relating the potential effects of data-gathering actions (history and exam questions, diagnostic tests, and trial therapy) to hypotheses (diseases and related conditions).³ The system has been supplied with a small amount of strategic knowledge, sufficient to tell it how to get the initial data on a patient (by asking questions) and how to gather evidence for hypotheses that have been triggered. It can therefore generate a set of possible actions that it might take to investigate the suspected diseases of the current patient. But it doesn't know *which* action to take next.

³The substantive knowledge was previously acquired for the MUM system [8].

Knowledge Bases
 INTERNET
 MUI
 System RIFs

ASK

Knowledge Acquisition
UMASS - EKSL

Copyright (c) 1987 Thomas Gruber

Objects Relevant to the Control Decision

EKG
 Applicability: APPLICABLE
 Classes: DIAGNOSTIC-TESTS.
 Cost: LOW
 Executed?: NO
 Network-dependents: DAF-EKG-FOR-UNSTABLE-ANGINA, DAF-EKG-FOR-CLAS
 Potentially-triggered: none.
 Value: unknown.

STRESS-TEST
 Applicability: APPLICABLE
 Classes: DIAGNOSTIC-TESTS.
 Cost: MEDIUM
 Executed?: NO
 Network-dependents: DAF-STRESS-TEST-FOR-PRINZ-METAL, DAF-STRESS-TE

COST of EKG
 display slot
 select as justification
 show value in full

Summary of Control State

Current Goals

GATHER-EVIDENCE-FOR-DIFFERENTIAL
 posted by PROPOSE-DIAGNOSTIC-EVIDENCE

Chosen Actions

BARRUM-SWALLOW
 proposed by PROPOSE-DIAGNOSTIC-EVIDENCE
 preferred by PREFER-CONCLUSIVE-EVIDENCE: conclusive evidence

EATING-PAIN-RELIEVED-BY-POSITIONAL
 proposed by PROPOSE-DIAGNOSTIC-EVIDENCE
 preferred by PREFER-CHEAP-DIAGNOSTIC-EVIDENCE: cheap evidence

EKG
 proposed by PROPOSE-DIAGNOSTIC-EVIDENCE
 preferred by PREFER-CONCLUSIVE-EVIDENCE: conclusive evidence

EPISODE-ACCOMPANIED-BY
 proposed by PROPOSE-DIAGNOSTIC-EVIDENCE

Justifications for the Current Control Decision

The POTENTIALLY-CONCLUSIVE-EVIDENCE of DIFFERENTIAL includes EKG.
 The POTENTIALLY-CONCLUSIVE-EVIDENCE of DIFFERENTIAL includes STRESS-TEST
 The COST of EKG is LOW.
 The COST of STRESS-TEST is MEDIUM.

Execution history

Acquired VALUE of SEX...MALE
 Acquired VALUE of AGE...50
 Acquired VALUE of CHEF-COMPLAINT...PAIN
 Acquired VALUE of CHEF-COMPLAINT-GENERAL-LOCATION...CHEST
 Acquired VALUE of CHEF-COMPLAINT-LOCATION...SUBSTERNAL
 Acquired VALUE of CHEF-COMPLAINT-DURATION...#.(duration 5 minutes)
 Acquired VALUE of EPISODE-INCITED-BY...(EATING STRESS)
 Acquired VALUE of EPISODE-PAIN-QUALITY...VISE-LIKE
 Acquired VALUE of EPISODE-RELIEVED-BY...RELIEVED-BY-ITSELF
 Acquired VALUE of EPISODE-TRIGGER-POINT...YES

Knowledge Acquisition Dialog

Why did you choose EKG over STRESS-TEST?

MU Output Window

more above

[ASK] To distinguish action EKG from STRESS-TEST,
 Specialize rule PREFER-CONCLUSIVE-EVIDENCE
 to match EKG but not STRESS-TEST.
 [ASK] Ways to specialize rule PREFER-CONCLUSIVE-EVIDENCE:
 add (IS LOW (COST ?ACTION))

Indicate that this feature is relevant to the current control decision.
 Clicking here will cause the statement of this fact to appear in the list of justifications.

Figure 1: A snapshot of ASK's justification interface.

Text in the windows is mouse sensitive; by clicking on words or lines, the expert can access to menus of operations on the entities represented by the text. The expert justifies a choice of an action by selecting features of relevant objects, which adds sentences to the contents of the justifications window. The justification interface shown in this screen is integrated with a general knowledge base browsing facility; windows and graphic displays for that facility are accessible to the user but not shown here.

The expert has selected a case in which the patient has symptoms that suggest various kinds of angina, and the knowledge system has run to the point where an interesting strategic decision must be made. The expert wants to teach the system how to choose its next action in this situation.

Eliciting justifications for a strategic choice

ASK: In this situation, I would choose one of:

EKG, Stress-test, Vasodilator-Tx, Angiogram.

The existing strategic knowledge causes the chest pain system to propose these four actions, but it cannot decide among them.

User: I would choose EKG.

ASK: Why? They are all equivalent to me.

Please distinguish EKG from one of the others.

ASK analyzes the current state of the strategic knowledge and determines that in order to account for the expert's choice, EKG, it should learn a rule that distinguishes EKG from the other actions.

User: I would prefer EKG over Stress-test because

"EKG potentially-confirms classic-angina."

The expert justifies the choice of EKG by selecting from facts about the current situation and features of EKG. The relation between EKG and one of the active hypotheses, classic-angina, is a fact computed dynamically by the inference net. The set of possible facts are provided by ASK in a browsing interface, and the expert searches through the possible facts to choose those that are "relevant to the current strategic decision." The selected facts serve as justifications for choosing EKG in this situation.

ASK: "potentially-confirms" does not distinguish EKG from stress-test.

Please provide further justifications for choosing EKG.

The potentially-confirms relation also happens to hold for the other three actions; they all potentially provide potentially conclusive evidence for hypotheses on the differential.

User: I would prefer EKG because "The cost of EKG is low."

With this justification, ASK can form a rule that distinguishes EKG from Stress-test. The rule says, roughly, "In this situation, select actions that are potentially-confirming and have a low cost." This rule reduces the set of possible next actions.

ASK: If I consider "potentially-confirms" and "cost", I would choose EKG and Vasodilator-Tx. Is this an improvement?

User: Yes.

ASK: Would you still choose EKG over Vasodilator-Tx?

If so, please justify.

ASK still needs more justifications to distinguish EKG.

User: Yes, because of the time-required.

Acquiring a new feature

ASK: Please define "Time-Required". I'll Help you...

ASK does not know about the term "time-required," and so it helps the expert define it as a new feature.

ASK: What kind of feature is time-required? Is it an attribute of actions, (like "cost"), a class of actions (like "diagnostic-tests"), a new object (like "the-differential"), an inferential value computed by rules (like "level-of-support"), or a dynamic relation (like "potentially-confirms").

User: attribute.

ASK plays the role of knowledge engineer, mapping the expert's construct to the representational primitives used in the knowledge base. This question about the feature type of the new feature can be difficult to understand when couched in implementation-level terms such as "dynamic relation," so ASK provides instances from the existing knowledge base as examples.

ASK: EKG is a member of the classes Diagnostic-tests, Data, and Actions. To which of these classes of objects will the attribute Time-required apply?

User: Diagnostic-tests.

The domain of a feature is a class. Since the expert is defining this term to describe the object EKG, ASK provides a list of its classes to choose from. Again, the context constrains the possible answers,

ASK: What possible values might time-required take? (The possible values of "Cost" are "free, cheap, low, ...")

User: immediate, few-minutes, an-hour, few-hours, a-day, few-days, ...

ASK: Is there an ordering over the values, such as "Immediate \leq few-minutes \leq an-hour ..."

After the new feature is well-defined, the ASK provides an interface for specifying the value of "Time-required" on the members of the class "diagnostic-tests" (or letting them inherit the default value).

When the expert is finished, the knowledge base has a new feature that can serve as a justification for selecting actions---in this case, it distinguishes EKG from Vasodilator-Tx.

Formulating and verifying a new strategy rule

ASK: Considering "potentially-confirms", "cost", and "time-required", I would choose EKG, too. I would paraphrase your advice as:

Select an action if:
a goal is to gather-evidence-for-differential, and
a hypothesis is in the-differential, and
the action potentially-confirms the hypothesis, and
the cost of the action \leq low, and
the time-required of the action = few-minutes.

Is this what you mean?

User: Yes.

ASK has formulated a new strategy rule based on the expert's justifications for choosing EKG. It is a specialization of the rule that proposed the original four actions. It maps the situation identified by the goal and the state of the differential to an equivalence class of acceptable actions—those that match the last three clauses. This rule causes the system to choose low cost, quick, potentially confirming actions when the goal is to gather evidence for hypotheses on the differential. Note that the rule captures a tradeoff : it says that actions that cost "low" (which is higher than "free" and "cheap"), are acceptable if they are potentially confirming and only take a few-minutes to run. The Vasodilator-Tx action can take weeks.

Generalizing the new rule

ASK: What if the time-required of EKG were immediate?
Would you still choose it?

User: Yes.

The new rule is operational but specific to actions that are very similar to the training example. ASK asks hypothetical questions to try to generalize the rule to cover actions that are near misses to EKG. Since it doesn't have any common sense, ASK has to ask whether it is better if tests take less time.

ASK: What if the time-required of EKG were an-hour?

User: Yes.

ASK: What about few-days?

User: No.

ASK tries to generalize the rule by making it match more actions. In this example, ASK checks for thresholds that would bound acceptable values of the time-required feature. The rule is generalized to apply to actions that require times from "immediate" to "an-hour."

1.4 Summary of the dialog session

In this dialog, ASK acquired strategic knowledge about how to choose among diagnostic tests and therapies. It elicited justifications for action from the expert, in the

by the inference net). *Filter rules* represent general constraints on the applicability of actions. A simple filter rule from the medical domain is "Filter actions that are contraindicated for the current patient."⁴ *Selection rules*⁵ represent the characteristics of actions that make them appropriate for a given strategic situation; each selection rule specifies an equivalence class of acceptable actions. A selection rule learned in the example dialog is

```
IF (in gather-evidence-for-differential (current-goals))
    (in ?hypothesis (the-differential))
    (in ?action (potentially-confirmed-by ?hypothesis))
    (<= (cost ?action) low)
    (<= (time-required ?action) few-minutes)
THEN (select ?action)
```

This rule says, when trying to get evidence for hypotheses on the differential, select actions that could potentially confirm hypotheses and that do not cost much or take much time.

At each iteration of the control cycle, strategy rules are run in a sequence: first, focus rules produce a set of possible actions; next, filter rules eliminate those that violate constraints; and then selection rules choose subsets of the remaining actions that are appropriate for the current situation. The union of selected subsets are recommended for execution.⁶ Within each category, rules are matched in an order specified by a precedence relation called *shadows*, which is roughly equivalent to a partial order based on generality.⁷ If a rule succeeds, then the rules that it shadows are pruned. A family of selection rules, in which the more specific rules shadow the more general, has the effect of choosing those actions judged to be acceptable by the most specific (most constraining) rules.

If the set of recommended actions is empty, then the system halts, and enters a

⁴Whether a particular action is contraindicated is computed by rules in the inference network relating patient characteristics with therapeutic actions).

⁵In previous reports, they were called "preference rules," but the term seems to have confusing connotations.

⁶If no selection rules apply, then all of the proposed rules are recommended. This allows the system to run with an incomplete knowledge base of selection rules.

⁷A rule is more general than a second rule if it tests a subset of the control features that the second rule tests or if it is less restrictive on the same control features. For example, a selection rule testing the condition "time-required = few-minutes" is more specific than the same rule with the condition "time-required <= an-hour." Precedence between two rules that are not ordered by generality can be specified explicitly (in the knowledge base); this is necessary to resolve conflicts between tradeoffs. For instance, if it is better to choose actions with low cost and high risk over actions with moderate cost and moderately high risk, then the selection rule representing the former tradeoff should shadow the rule for the latter.

knowledge acquisition dialog (since it has insufficient strategic knowledge to continue).⁸ If the chosen set of actions is a singleton, then the system executes that action. Otherwise an action is picked arbitrarily (e.g., by the user or randomly), and executed. Thus the performance system can run with *partial* strategic knowledge, as long as the focus rules can propose a sufficient set of actions to solve the problem.⁹

2.2 Eliciting justifications and new features from experts

ASK's method for eliciting justifications is based on a correspondence between the expert's justifications for choosing actions and the control features used by strategy rules: attributes, functions, and relations of objects in working memory. Justifications are facts about the current situation—instantiations of control features on objects in working memory. The expert justifies the choice of an action by identifying those facts that are relevant to the decision. ASK provides a browsing interface (something like a friendly Lisp Inspector) for examining the state of working memory to find relevant facts. The expert searches for relevant objects (e.g., actions, hypotheses, goals) and the relevant control features that apply to them (cost of actions, confirming evidence for hypotheses), and indicates which facts serve as justifications. ASK uses heuristics to guide this search (by priming windows with suggested values). By clicking on the "cost" feature of the EKG object, the expert causes the statement "The cost of EKG is low" to appear in the justifications window. With a mouse-and-menu interface the expert manipulates the contents of the justifications window until ASK is satisfied (i.e., the set of justifications distinguishes EKG). Figure 1 shows the state of the screen during a justification session.

If the available control features are inadequate, the expert can define a new feature. ASK provides a custom frame editor to help the expert define a new feature, using the context of the current object and existing features in the knowledge base to guide the process.

2.3 The Learning Cycle

ASK learns strategy rules from the expert's justifications for particular strategic decisions. The learning is structured in the dialog cycle, shown in figure 3, consisting of the following operations.

1. **Credit assignment** determines what part of the system's knowledge needs to be changed, and how. The expert runs a case through the knowledge system, setting up an example where the system's performance (i.e., the actions recommended

⁸Halting conditions are represented by strategy rules that recommend a special halt action.

⁹In the absence of focus rules, all actions that the system knows about are considered at each iteration.

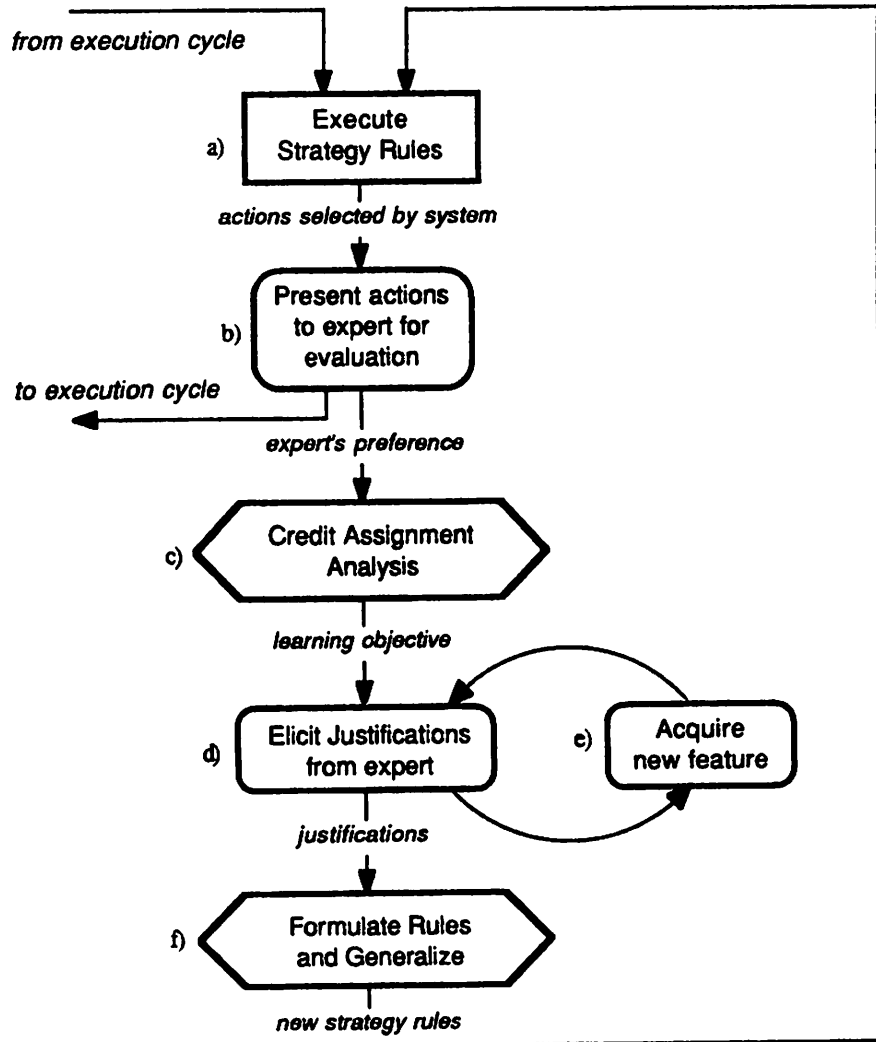


Figure 3: The ASK Knowledge Acquisition Dialog Cycle

in a cycle) differs from the expert's. ASK analyzes the discrepancy between the expert's and system's choices, and then examines how the current set of strategy rules had matched. The output of credit assignment analysis is a specific learning objective, such as to specialize a particular selection rule so that it matches the expert's choice and does not match an action that the system suggested.

2. **Elicitation of justifications.** ASK helps the expert enter justifications, which it interprets as tests on control features, until it can formulate an expression that satisfies the learning objective.
3. **Formulating strategy rules.** ASK uses the justifications to form conditions on the left hand side of a new strategy rule. For example, if the expert says that the cost of the preferred action is relevant, ASK formulates a test such as `(= (cost ?action) low)`. Rules may be formed "from scratch," consisting entirely of clauses from the justifications, or may be refinements of existing rules, with additional clauses.¹⁰
4. **Generalization.** By formulating new rules in terms of variables, ASK is doing constant-to-variable induction. ASK also generalizes strategy rules with syntactic induction operators such as dropping conditions and extending intervals. ASK always asks the expert to verify the inductive generalizations it makes. monitoring a It asks for verification of the constant-to-variable generalization by paraphrasing the rule, and queries other forms of generalization by posing hypothetical variants on the training case.

3 Why It Works: Sources of Power and Limitations

ASK is a synthesis of ideas and techniques from several areas of work in AI. At one level it serves to validate and generalize, demonstrating the power of existing ideas in a new problem. In addition, the design decisions upon which ASK is built locate a point in a space of AI programs, with particular advantages and compromises. The following sources of power in ASK illustrate the payoffs and tradeoffs.

Interactive acquisition in the context of problem solving

ASK is a TEIRESIAS-style knowledge acquisition assistant, and reaffirms many of Davis' conclusions. It engages the expert in a dialog in the context of a running performance system, and it uses knowledge about the structure and behavior of its rules at run time to do credit assignment. Similar experiments in learning search

¹⁰A shadowing link is added between the new rule and the rule it refines (specializes).

strategy from people's critiques of performance systems have been run in SOAR [19] and BBL [25]. The context of an actual case and the internal state of the performance system provide powerful constraints on the knowledge acquisition problem—a far cry from “Tell me what you know!”

Task-specific architecture

ASK is a *task-specific* knowledge engineering tool, in that it specializes in acquiring a particular kind of knowledge and is integrated with an architecture which supports the task [21,30]. The strategy rule representation, the technique of eliciting justifications, and the learning dialog are all designed around the peculiar problems of acquiring strategy. The power of the task-specific approach has been demonstrated by tools that acquire knowledge for other tasks, for instance: hierarchical classification (ETS [2]), heuristic classification (MOLE [17]), hardware troubleshooting (ASTEK [26]), iterative design/configuration (SALT [29]), and the management of cancer therapy (OPAL [33]).

Explicit, declarative representation of control

Several researchers have cited the advantages of representing control knowledge explicitly, for explanation, software engineering, and knowledge acquisition purposes [6,21,23,34,37]. If the control knowledge of a program is embedded in a procedure, or worse yet, implicit in priorities and rule orderings, then it is difficult for people and programs to analyze, explain, and maintain it. Because the basis for control is explicit rules, ASK can reason about their behavior (for credit assignment), and can make use of strategic knowledge in a form that experts can provide (i.e., justifications). The idea of representing control knowledge as explicit rules was introduced by Davis [11], and was developed as a representation of strategy by Clancey [7]. The productions that create preferences in SOAR [27] formulate control decisions with rules in a manner similar to ASK. In every case, the rule formalism is used because it makes the knowledge about what to do next *explicit*, so that it can be reasoned about by the system.

Simple correspondence between structure and function

Credit assignment analysis and generalization are based on the correspondence between the clauses of strategy rules and their behavior. If the system failed to propose an action desired by the expert, the culprit is a missing focus rule. If the system failed to discriminate actions, credit is given to an overly-general selection rule or missing filter rule (generality is determined syntactically). Contrast this analysis with the knowledge that a programmer brings to bear when debugging the control flow of a program, in which the structure of the code does not correspond neatly to its behavior.

Small-grained control and local decisions

Since experts do not need a general control algorithm to do their jobs, they presumably have little experience in describing their behavior that way; however, they can say what they do in real cases. ASK shows that strategic knowledge can be acquired from experts if it is elicited in specific situations and then generalized. This is possible because strategy rules model iterative, local control decisions. SALT also acquires knowledge about what to do next (when putting things together under constraints), and it too is based on a model of small-grained, iterative control with backtracking [29].

Generalizable formalism

Strategy rules can represent levels of generality of strategic knowledge. The more specific strategy rules are more constraining—they choose fewer actions for more reasons. The generalizability of strategy rules facilitates knowledge acquisition in two ways. First, the developing knowledge system can run without complete strategic knowledge; it falls back on the more general strategy rules. This allows the knowledge engineer to write a few general strategy rules and the expert to “fine-tune” the strategy by specializing them. Second, generalizability of the representation allows ASK to use simple syntactic generalization operators for inductive learning. Similarly, the hierarchical structure of the ACCORD control language [24] enables a system called WATCH [22] to generalize search strategies from examples.

Reformulating strategy as classification

Strategy is naturally thought of in procedural terms, because that is how programs know what to do next. But representing strategy in procedures complicates induction. The problem of learning sequences from examples, much less arbitrary procedures with internal state, is very hard [15]. On the other hand, the problem of learning classification rules is well understood [31]. ASK reformulates the problem of “how to act” as mappings between situations and sets of actions (i.e., classification rules), and as a consequence can use existing learning techniques. This is old and powerful technique. Several systems for learning search control knowledge represent control with productions (e.g., LP [37] and SOAR [27]). Waterman formalized a production system for the explicit purpose of learning strategy; in his Poker Player, productions mapped heuristic descriptions of the state of the game to actions such as “bet” [39].

Learning by asking

ASK learns “at the knowledge level” [13] with a human in the loop, by integrating elicitation and induction techniques. The expert is an integral player in learning cycle, selecting the examples and defining the description language. Both roles are strong sources of bias [35], [38]. The expert also serves as critic [14], and the knowledge system is the performance element. ASK performs the credit assignment and generalization, and manages the dialog. ASK employs a weak form of the *single representation trick* [12]—the assumption made in some learning programs that the training data are in the same form as the concept description—by mapping experts’ justifications to control features in strategy rules. The *new term problem*—that learning systems cannot learn what they cannot represent—is not a problem for ASK, again because the description language is extended by the user. The human-machine synergy that results from integrating elicitation and learning techniques has also been explored in EURISKO [28], AQUINAS [3], MOLE [16], BLIP [32], and PROTOS [1].

4 Major Limitations

The class of problems to which ASK applies

ASK only applies to problem solving tasks with these characteristics: expert strategy is important; the strategy can be viewed as small-grained iterative control; the set of possible actions can be described as objects with attributes and relationships with other objects; and the set of control features that are relevant to a given strategic situation is small. Positive examples include prospective diagnosis in medicine and toy robot control (that can be modeled with productions). An example of a problem that does *not* fit ASK’s model (i.e., a near miss) is cancer therapy management, the domain of ONCOCIN [36]. Experts in cancer treatment apply strategic knowledge to generate *protocols*: long, detailed, branching plans (sequences of therapeutic actions). We found that the justifications for choosing the next action in protocols were often inter-action dependencies of the form “because I just did action-X and it had this result,” rather than “because action-Y has the appropriate properties for this patient in this situation.” It seems that many of the features of the situation that determine which action should go next are really descriptions of immediate past history. Although one could certainly refer to the action history as a justification and generate strategy rules, they would not generalize well, because the terms would be constants (specific actions). This is reflected in a difference in problem solving methods; ONCOCIN’s designers chose a skeletal plan refinement method, while ASK uses the reactive style of planning.¹¹

¹¹And a tool built specifically for ONCOCIN’s method—OPAL—has been quite successful at acquiring protocols [33].

Multiple experts

One of the motivations for separating strategic and substantive knowledge is the hope that the the strategy could be used in several systems within a domain. By acquiring a single expert's strategy, we may be biasing the resulting system to use an idiosyncratic approach. If two or more experts work on the same knowledge system, their strategic knowledge may differ significantly. ASK has no mechanism to merge the recommendations of several experts, as is found in AQUINAS [3].

The substantive knowledge must be acquired and correct

ASK builds upon an existing substantive knowledge base, relying on the objects and features already defined. It is assumed that the substantive knowledge can be acquired, possibly with a tool for classification, such as ETS or MOLE. A more serious problem is the assumption that the substantive knowledge is correct. ASK's credit assignment algorithm cannot handle bugs in the substantive knowledge base, and they are certain to occur. The user must be able to recognize this problem when it occurs.¹²

Dependence on knowledge engineers

Although ASK provides a helpful interface for defining new features, some require programming to implement. These include functions which compute dynamic evidential relationships between actions and hypotheses; they must be implemented by knowledge engineers. And of course ASK cannot be used in isolation to implement a knowledge system; its speciality is the strategic component. Yet like ETS [2], ASK can serve the purpose of identifying features that are relevant to an expert's domain, without the intervention of a knowledge engineer.

5 Conclusion

ASK's design is a tightly coupled integration of an operational representation, an elicitation technique, and a learning method. The representation is constrained by limitations on what can be elicited from experts. The learning method depends on the expert for examples and bias, and it relies on properties of the representation

¹²A surprising side-effect of ASK's sensitivity to the substantive knowledge base is its usefulness (to a knowledge engineer) as a testing vehicle. A bug in a control feature, for example, might cause the "wrong" strategy rule to fire or fail to fire, producing a bad choice of action. If that control feature is given as the justification for choosing the action, ASK will complain that it needs more justifications for distinguishing an action that should have been distinguished—but had a faulty control feature. The ODYSSEUS learning apprentice exploits this sensitivity of metarules to bugs in the metarelations; it assumes correct strategic knowledge and uses it as a model to acquire substantive knowledge [40].

for generalization. Overall, the sources of power and the limitations we have discussed result from a design which balances expressive power with the objectives for automating the acquisition of strategic knowledge from experts.

References

- [1] E. R. Bareiss, B. W. Porter, and C. C. Wier. Protos: An exemplar-based learning apprentice. In *Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-based Systems Workshop*, Banff, Canada, October 1987. To appear in the *International Journal of Man-Machine Studies*.
- [2] J. H. Boose. *Expertise Transfer for Expert System Design*. Elsevier, New York, 1986.
- [3] J. H. Boose and J. M. Bradshaw. Expertise transfer and complex problems: Using AQUINAS as a knowledge acquisition workbench for expert systems. *International Journal of Man-Machine Studies*, 26(1):21-25, January 1987.
- [4] D. Chapman and P. E. Agre. Abstract reasoning as emergent from concrete activity. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions and Plans, Proceedings of the 1986 Workshop at Timberline, Oregon*, pages 411-424, 1987.
- [5] W. J. Clancey. Acquiring, representing, and evaluating a competence model of diagnosis. KSL Memo 84-2, Stanford University, February 1984. To appear in Chi, Glaser, and Farr (Eds.), *Contributions to the Nature of Expertise*, in preparation.
- [6] W. J. Clancey. The advantages of abstract control knowledge in expert system design. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 74-78, Washington, D.C., August 1983.
- [7] W. J. Clancey. Representing control knowledge as abstract tasks and metarules. In M. Coombs and L. Bolc, editors, *Computer Expert Systems*. Springer-Verlag, forthcoming. Also KSL 85-16, Stanford University.
- [8] P. R. Cohen, D. S. Day, J. Delisio, M. Greenberg, R. Kjeldsen, D. Suthers, and P. Berman. Management of uncertainty in medicine. *International Journal of Approximate Reasoning*, 1(1):103-116, 1987.
- [9] P. R. Cohen, M. Greenberg, and J. Delisio. MU: A development environment for prospective reasoning systems. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 783-788, Seattle, Washington, July 1987.

- [10] R. Davis. *Applications of meta-level knowledge to the construction, maintenance, and use of large knowledge bases*. PhD thesis, Computer Science Department, Stanford University, 1976. Reprinted in R. Davis and D. B. Lenat (Eds.), *Knowledge-Based Systems in Artificial Intelligence*, New York: McGraw-Hill, 1982.
- [11] R. Davis. Meta-rules: Reasoning about control. *Artificial Intelligence*, 15(3):179-222, 1980.
- [12] T. G. Dietterich. Learning and inductive inference. In P. R. Cohen and E. Feigenbaum, editors, *The Handbook of Artificial Intelligence, Volume III*. William Kaufmann, Inc., 1982.
- [13] T. G. Dietterich. Learning at the knowledge level. *Machine Learning*, 1(3):287-316, 1986.
- [14] T. G. Dietterich and B. G. Buchanan. The role of the critic in learning systems. Report STAN-CS-81-891, Department of Computer Science, Stanford University, December 1981.
- [15] T. G. Dietterich and R. S. Michalski. Learning to predict sequences. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach, Volume II*, volume 2, pages 63-106. Morgan Kaufmann Publishers, Inc., Palo Alto, Ca., 1986.
- [16] L. Eshelman. MOLE: A knowledge acquisition tool that buries certainty factors. In *Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-based Systems Workshop*, Banff, Canada, October 1987. To appear in the *International Journal of Man-Machine Studies*.
- [17] L. Eshelman and J. McDermott. MOLE: A tenacious knowledge acquisition tool. *International Journal of Man-Machine Studies*, 26(1):41-54, 1987.
- [18] R. J. Firby. An investigation into reactive planning in complex domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 202-206, Seattle, Washington, 1987.
- [19] A. Golding, P. S. Rosenbloom, and J. E. Laird. Learning general search control from outside guidance. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 334-337, Milan, Italy, 1987.
- [20] T. R. Gruber. Acquiring strategic knowledge from experts. In *Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-based Systems Workshop*,

Banff, Canada, October 1987. To appear in the *International Journal of Man-Machine Studies*.

- [21] T. R. Gruber and P. R. Cohen. Design for acquisition: principles of knowledge system design to facilitate knowledge acquisition. *International Journal of Man-Machine Studies*, 26(2):143-159, 1987.
- [22] J. Harvey. WATCH - inductive learning of control abstractions. Digital Equipment Corporation, Palo Alto, California.
- [23] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251-321, 1985.
- [24] B. Hayes-Roth, A. Garvey, M. Johnson, and M. Hewett. A layered environment for reasoning about action. Technical Report KSL 86-38, Computer Science Department, Stanford University, April 1986.
- [25] B. Hayes-Roth and M. Hewett. Learning control heuristics in a blackboard environment. Technical Report HPP-85-2, Computer Science Department, Stanford University, 1985.
- [26] C. Jacobson and M. J. Freiling. ASTEK: A multi-paradigm knowledge acquisition tool for complex structured knowledge. In *Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-based Systems Workshop*, Banff, Canada, October 1987. To appear in the *International Journal of Man-Machine Studies*.
- [27] J. E. Laird, A. Newell, and P. S. Rosenbloom. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33:1-64, 1987.
- [28] D. B. Lenat. EURISKO: A program that learns new heuristics and domain concepts. *Artificial Intelligence*, 21:61-98, 1983.
- [29] S. Marcus. Taking backtracking with a grain of SALT. *International Journal of Man-Machine Studies*, 26(4):383-398, 1987.
- [30] J. McDermott. Making expert systems explicit. In *Proceedings of the Tenth Congress of the International Federation of Information Processing Societies*, Dublin, Ireland, 1986.
- [31] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine Learning: An Artificial Intelligence Approach*. Tioga Press, Palo Alto, Ca., 1983.
- [32] K. Morik. Acquiring domain models. *International Journal of Man-Machine Studies*, 26(1):93-104, 1987.

- [33] M. A. Musen, L. M. Fagan, D. M. Combs, and E. H. Shortliffe. Using a domain model to drive an interactive knowledge editing tool. *International Journal of Man-Machine Studies*, 26(1):105, 1987.
- [34] R. Neches, W. Swartout, and J. Moore. Enhanced maintenance and explanation of expert systems through explicit models of their development. *Transactions on Software Engineering*, SE-11(11):1337-1351, 1985.
- [35] E. L. Rissland. The problem of intelligent example selection. In *Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-based Systems Workshop*, Banff, Canada, October 1987. To appear in the *International Journal of Man-Machine Studies*.
- [36] E. H. Shortliffe. ONCOCIN: An expert system for oncology protocol management. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 876-881, Tokyo, Japan, 1981.
- [37] B. Silver. *Meta-level Inference: Representing and Learning Control Information in Artificial Intelligence*. North-Holland, New York, 1986.
- [38] P. Utgoff. *Machine Learning of Inductive Bias*. Kluwer Academic, Norwell, MA, 1986.
- [39] D. A. Waterman. Generalization learning techniques for automating the learning of heuristics. *Artificial Intelligence*, 1:121-170, 1970.
- [40] D. C. Wilkins, W. J. Clancey, and B. G. Buchanan. Knowledge base refinement by monitoring abstract control knowledge. *International Journal of Man-Machine Studies*, 27:281-293, 1987.