Reasoning with Incomplete Knowledge in a Resource-Limited Environment: Integrating Reasoning and Knowledge Acquisition

Mark S. Fox

CMU-RI-TR-81-3

Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

27 March 1981

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-78-C-1551.

This paper appeared in the *Proceedings of the Seventh International Conference on Artificial Intelligence*, Vancouver, BC, Canada, Aug. 1981.

Table of Contents

List of Figures

ABSTRACT

This paper describes an approach to reasoning with incomplete information in a resource-limited environment. Approaches to date either assume infinite resources and proceed to enumerate a large inference space, or assume few resources and ignore the missing information. They do not reason about resource constraints and the inference methods admissible under them. A HEARSAY-II-like system is described where each knowledge source is a separate production system. During rule evaluation, a rule antecedent is evaluated using minimal-resource methods. A rule antecedent is evaluated to true, false, or an expected resource cost to acquire the information necessary to complete its evaluation. If conflict resolution chooses a partially evaluated rule, it posts a goal asking other knowledge sources to provide the missing information, suspends the knowledge source, and informs the knowledge source's manager about the suspension and accompanying goal. The manager decides whether the goal is worth pursuing now, the amount of resources to apply to the task, what knowledge source to apply, and when to give up. The knowledge sources that attempt the goal can implement a variety of inferential and knowledge acquisition techniques.

1. INTRODUCTION

The purpose of this paper is to describe a metro of reasoning with incomplete information in rule-based systems. In particular, it describes how an "expert program" can gracefully integrate the dynamic acquisition of knowledge with its reasoning process. The integration process allows the system to reason about the most appropriate methods for acquiring knowledge. The primary representation of knowledge in rule-based systems is the production rule (Newell & Simon, 1972; Davis & King, 1975). A production rule is of the form:

Antecedent ---> Consequent

The antecedent can be any expression which if true upon evaluation results in the evaluation (execution) of the consequent. The antecedent is classically a pattern to be matched against data in memory, but can be as complex as a first-order predicate expression or an arbitrary LISP expression. A basic assumption in rule-based systems is that the antecedent is a total-function which evaluates to true or false. In a rule system where the antecedent is a pattern to be matched, the assumption holds; the pattern matches something in memory or it does not. But the failure of a match can arise from two sources, either the pattern does not exist in memory (is absolutely or temporarily false), or memory lacks sufficient information to determine truth or falsity, i.e., the knowledge base is incomplete. In the latter case, an unsuccessful match should be interpreted as "I don't know" instead of false. While not new, this problem has many names such as partial-matching (Hayes-Roth, 1978), partial-information inferencing (Joshi, 1978), and backward chaining (Shortliffe, 1975). More generally, it is the problem of reasoning with incomplete information.

In a rule-based system the control system evaluates each rule antecedent and chooses one rule from those that are true. But it is often the case that pattern antecedents partially match memory, and for predicate antecedents a subset of predicates evaluate to true while others are not disprovable. The question is: "What rule should be chosen to execute?" Previous solutions to this problem fall into a continuum where one end uses maximal knowledge but requires large amounts of resources to learn, infer or acquire the missing information, while the other end uses little knowledge and minimal resources. In particular, the backward chaining of Mycin (Shortliffe, 1976) does a complete search of the rule space looking for rules which may provide the missing information required to evaluate an antecedent. Klaht (1978) reduces the search cost of Mycin's backward chaining by using rule abstractions to plan the backward chaining of rules. Abstraction reduces the information available. In both these approaches, the rules determine what information is relevant and the method of acquisition. Joshi (1978) decreases the cost of choosing a rule even further by not searching for the missing information. Instead, a partial match metric based on specificity and certainty is derived and the highest valued rule is chosen. McDermott (1978) deals with partial-matches by creating a separate rule for each sub-pattern of the antecedent which is of use to the problem-solving. Rosenberg (1979) augments the Planner system with meta-theorems that propose alternate ways of supporting a goal when existing theorems fail. Mostow (1977) has proposed the specification of how, what, why and when information for each rule, to guide in the selection of rules and the gathering of information of discrimination amongst them. In all these approaches, the more resources expended, the more complete the search, the more certain is the outcome. The important questions here are: What is the level of certainty required by the problem-solving, and what is the (resource) cost of reducing uncertainty?

2. Reasoning with Incomplete Knowledge in a Resource-Limited Environment

Reasoning, problem-solving, and learning have all been characterized as heuristic search. In the more interesting domains, the search space is so large that it cannot be completely searched, hence many of the problems center around reducing the size of the search space. Not being able to circumnavigate the search space is basically a resource limitation, e.g., time and space. Under such circumstances, reasoning programs must efficiently assign and use resources to solve a problem. Hence, rule-based systems, when faced with the inability to evaluate an antecedent, must balance adequacy of performance against resource limitations. Lenat et al. (1979) have discussed the problem of reasoning in resource-limited environments under the name of cognitive economy, and Lenat (1976) has used the schema of assigning time slices to rules to limit resource expenditures.

For example, if there exists a rule of the following form:

```
    IF (X ∈ Vertebrata) THEN .....
```

and X in this case is a lion, then there are a number of ways of evaluating the antecedent "X is a member of Vertebrata":

1. Is X equal to Vertebrata?

2. Search the is-a hierarchy of the knowledge base. Lion --> Panthera --> Felidue --> Carnivora --> Mamalia --> Vertebrata.

3. Look for other inference rules that may provide the answer.

4. Match the lion schema against the Vertebrata schema.

5. Ask the human sitting at the terminal whether it is true.

6. Assume it's true and retract if it leads to inconsistent results.

This example raises a variety of problems that can be found in almost any domain. There are many algorithms (methods) available for answering a question including learning and knowledge acquisition methods. Some require only a few resources such as the equality check, some require potentially a lot of resources such as the is-a search and the pattern match, while others require non-replenishable resources. For example, if a system can ask the user questions, then it is in the best interests of the system not to ask the user too many questions because he may become annoyed and provide fewer and less accurate answers.

When faced with one or more rules whose antecedent cannot be evaluated, the reasoning system must address the following problems:

- Should it continue to work at evaluating the antecedent?

- How many resources should be expended in evaluation?

- What methods should be chosen to evaluate the antecedent?

3. Meta-Evaluation: Resource Limited Evaluation

The KS system of Fox (1981a) is a discovery system which reasons with incomplete information. A simplified view of its program architecture is a distributed set of HEARSAY-II-like systems (Erman et al., 1980). That is, there are *departments* where each department has *employee* knowledge sources and a *manager* knowledge source. Departments can form hierarchies of heterarchies with communication channels and shared data spaces. Each knowledge source is a rule-based system with access to a global knowledge base. When a knowledge source is executing, each of its *visible* rules are *meta-evaluated.* Meta-evaluation attempts to evaluate an antecedent using minimal resources. If it cannot evaluate the antecedent successfully, it assigns to the rule an *expected cost* for successfully evaluating the rule using other methods. The conflict resolution algorithm for the knowledge source then chooses a rule based on specificity, relation to the knowledge source's goal, number of parts evaluated, and expected cost (if not successfully evaluated). If the antecedent of the rule chosen was evaluated successfully, then its consequent is evaluated and the process begins again. If the antecedent was assigned an expected cost, then the knowledge source is suspended and the portions of the rule not evaluated are placed as goals in the knowledge base for other knowledge sources to evaluate. Other knowledge sources' preconditions fire and instantiate them. Each knowledge source provides different methods at different costs. The manager knowledge source decides to which knowledge source to assign the task, and how many resources it will expend. A knowledge source may run out of resources before it finishes, leaving the manager to assign more resources or to assign the task (goal) to another employee. Once the antecedent is finally evaluated, the suspended knowledge source is awakened and passed the value by the manager. Processing of the knowledge source continues.

4. Meta-Evaluation: An Example

A rule antecedent in the KS system can be any arbitrary Lisp expression but includes existential and universal quantification as a basic knowledge base search mechanism. To simplify matters, we will assume a single department with a manager and three employees:

1. CONSPEC: knowledge source that creates new schemata via specialization (figure 1)[1]

2. INTROSPEC: primary inference knowledge source.

3. QUESTASK: knowledge source that acquires knowledge by asking the user a question.

[1]The schema depicted in figure 1 has the following form: A schema definition is surrounded by double braces "{{ <schema def> }}." A schema is composed of multiple views, each of which is surrounded by single braces "{ <view type> <viewed schema> <view slots> }." Each view has a set of slots defined (inherited) in that view. Each SLOT which is printed as SMALL CAPITALS, has a set of *facets* printed in *italics.* One of the facets is the *value* of the SLOT. When only the *value* facet has a filler, the SLOT and the filler are printed without naming the facet. A schema also has slots that are independent of its views. They are printed separately. Each slot can act as a bi-directional inheritance relation. Slots that act as inheritance relations are printed in the view format. See (Fox, 1981a, 1981b) for more information.

```
{{ Conspec
   {IS-A Employee
        MANAGER:  Example-Manager
        WINDOW:
                 Restriction:  (SET (TYPE is-a rule))
                 Value:  R1 R2
        STATE:
                 Restriction:  (OR rule-block mail-block completed
                                     failed ready)
                 Value:  ready
        PRECONDITION:
                 restriction: (SET (TYPE is-a capability))
        BUDGET:
        SPACES:
        MAIL-BOX:
        SUSPENDED-RULE: }
  }}
```

Figure 1: CONSPEC Schema

```
{{R1
    {IS-A Rule
         ANTECEDENT:  (X ∈ VERTEBRATA)
         CONSEQUENT: ...}
  }}

{{R2
    {IS-A Rule
         ANTECEDENT:  (X ∈ VERTEBRATA) AND
                    (X.Color IS-SIMILAR-TO(X.Environment).Color)
         CONSEQUENT:  ...}
  }}
```

Figure 2: CONSPEC Visible Rules

CONSPEC has two rules in its rule set (figure 2). Rule R1's antecedent tests whether X is a member of VERTEBRATA. Rule R2 is similar to R1 but has been specialized by the added condition "similar" which tests whether X's color is similar to the color of X's environment. For this example, X is bound to the schema LION.

4.1. Low-Cost Evaluation

The first phase of rule meta-evaluation is low-cost evaluation of functions in rule antecedents. For each function in the antecedent, a schema exists in the knowledge base that defines how it is evaluated. These function schemata come in three flavors:

1. *Low-cost Function*: uses bounded, small set of resources during execution. Contains the code to be executed.

2. *Variable-cost Function*: uses variable set of resources (possibly unbounded) in evaluation. Search based functions (e.g.,there-exists) are of this type. This schema contains the expected-cost of evaluation information and the goal name to be assigned to the function.

3. *Multi-Function*: a function that can be evaluated in more than one way, some low-cost and others variable-cost. An example is set membership ϵ. The schema contains the low-cost functions and the expected-cost and goal name of the variable-cost approach.

If the function is low-cost, then the code in the schema is executed. If the function is variable-cost, then the expected-cost for evaluation and the name of the goal to attach to the function is returned. If the function is a multi-function, then the low-cost functions are evaluated until one succeeds. If none succeed, then the expected cost and goal name is returned.

```
{{ ϵ
    { IS-A Multi-Function
            LOW-COST-FUNCTION:  ϵ-equal
                                ϵ-depth3
            EXPECTED-COST:  (seconds 30)
            GOAL-NAME:  ϵ-goal}
  }}
```

Figure 3: ϵ Multi-Function

```
{{ Lion
    {IS-A Panthera
         COLOR: tan
         ENVIRONMENT: Grassland}
  }}

{{ Grassland
    {IS-A Place
         COLOR: brown, green, tan}
  }}
```

Figure 4: Lion and Grassland Schemata in Memory

We will assume that in this example, there is enough information to answer rule R2's color test but none of the low-cost function tests for the multi-function ϵ succeed (figure 3). The first low-cost function ϵ-equal uses equality to test membership, but LION≠VERTEBRATA so it fails. The second low-cost function ϵ-depth3, searches to depth 3 up the IS-A hierarchy (Lion--> Panthera --> Felidue --> Carnivora), and also fails. Hence, the expected cost of (seconds 30) is returned for ϵ along with the ϵ-GOAL name to be assigned to the antecedent if the rule is chosen by conflict resolution.

4.2. Conflict Resolution

The rules R1 and R2 cannot be evaluated since lion and vertebrata are more than 3 levels apart; thus, for each rule, the expected-cost of (seconds 30) is returned. The conflict resolution system looks at the two rules and chooses R2 for three reasons: 1) R2 is more specific than R1; 2) answering R2 will also answer R1; and 3) R2 has more subparts successfully matched than R1. Cost, specificity, evaluability, and relation to employee's goal are used to choose a rule.

4.3. Goal Posting and Assignment

Each department has associated with it one or more sections of the knowledge base called spaces. Each employee's precondition is keyed to one or more of these spaces. If the chosen rule has an expected cost associated with it, the knowledge source is suspended (figure 5) and the unevaluated functions are posted in the department's goal space as schemata with the goal-names assigned and a state attribute of POSTED (figure 6). All the other "interested" employees then test the goal against their precondition (apply the contents of the test slot in each of the capability schemata that fill the precondition slot) and, if true, inform the manager of their applicability. In this case, both the INTROSPEC and QUESTASK employees fire and inform the manager that they can (possibly) evaluate the ϵ-GOAL. The manager then compares the possible approaches offered by the employees. INTROSPEC (figure 7) would use more resources than QUESTASK (figure 8), but the factor of bothering the human user is considered, lowering the overall utility of QUESTASK. INTROSPEC is chosen, assigned resources based on availability, expected cost and what are available, and is initiated. The state of the ϵ-GOAL schema in the goal space is changed from POSTED to ASSIGNED (figure 9).

```
{{ Conspec
    {IS-A Employee
        MANAGER: Example-Manager
        WINDOW: R1 R2
        STATE: suspended
        SUSPENDED-RULE: R2
        GOAL: ϵ-goal-1 }
 }}
```

Figure 5: Suspended CONSPEC Schema

```
{{ ϵ-goal-1
    {IS-A Goal
            GOAL: ϵ-GOAL
            PARAMETERS: Lion Vertebrata
            SUSPENDED-KS: Conspec
            RULE: R2
            STATE: posted
            EXPECTED-COST: (seconds 30)
            RESULT:
            CONTRACTED-KS:
            BUDGET: (seconds 50)
            ATTEMPTS: }
  }}
```

Figure 6: Posted ϵ-GOAL

```
{{ Introspec
    {IS-A Employee
         MANAGER: Example-Manager
         STATE: ready
         PRECONDITION: { IS-A Capability
                  GOAL: ϵ-GOAL
                  TEST: ϵ-GOAL-TEST-INTROSPEC
                  APPROACH: ϵ-Match-Rule-Set
                  COST: (seconds 25)}
         }
  }}
```

Figure 7: Introspec Schema

```
{{ Questask
    { IS-A Employee
         MANAGER: Example-Manager
         STATE: ready
         PRECONDITION: {IS-A Capability
                  GOAL: ϵ-GOAL
                  TEST: ϵ-GOAL-TEST-QUESTASK
                  APPROACH: ϵ-question
                  COST: (questions 5)}
        }
  }}
```

Figure 8: Questask Schema

```
{{ ε-goal-1
   { IS-A Goal
        GOAL: ε-GOAL
        SUSPENDED-KS: Conspec
        RULE: R2
        STATE: assigned
        EXPECTED-COST: (seconds 30)
        RESULT:
        CONTRACTED-KS: Introspec
        BUDGET: (seconds 50)
        ATTEMPTS: }
  }}
```

Figure 9: Assigned ε-GOAL

4.4. Goal Satisfaction

The INTROSPEC employee tries a variety of methods available in its rule set to solve the goal. For each attempt that fails, it records on the goal schema the attempt type and the amount and type of resources consumed (figure 10). This information is used by the system to update function schemata expected-costs and by the manager to decide what to do next if the employee fails at its task. If INTROSPEC succeeds, it changes the state of the goal schema to EVALUATED, informs the manager, and ends. The manager activates (puts in the ready queue) the suspended employee. If INTROSPEC fails or runs out of resources, the manager is informed and decides whether to continue pursuing the goal, what resources to expend, and to whom they should be assigned. In this case, if INTROSPEC fails, QUESTASK is initiated and the user is queried as to whether a lion is a vertebrata.

```
{{ ε-Attempt-1
   { IS-A Goal-Attempt
        KS: Introspec
        METHOD: ε-Match-Rule-Set
        COST: (seconds 20)
        REASON: <list of unmatched attributes>}
  }}
```

Figure 10: ε-Goal Attempt by Introspec Knowledge Source

5. Observations and Conclusions

We described an approach for reasoning with incomplete knowledge in a rule-based system which integrates reasoning and knowledge acquisition. Simply, it attempts to evaluate a rule's antecedent using resource-miserly methods. If these fail, then it posts the antecedent as a goal for evaluation by other knowledge sources in the system. This approach has several strengths:

1. Antecedents are quickly evaluated when the information to answer it exists. The cost of meta-

evaluation at this point is the cost of interpreting the function's schema -- small for a LOW-COST FUNCTION, more expensive for a MULTI-FUNCTION.

2. When information is incomplete, a variety of methods can be brought to bear on evaluating the antecedent. These methods require various amounts of resources.

3. The choice of method to apply is left to the manager knowledge source which has a more global view of the problem, hence can intelligently choose the best approch and the resources to assign.[2]

4. Even if an antecedent cannot be evaluated, the manager or an employee can force it to be true or false, in essence, creating an hypothesis requiring dependency analysis capabilities (London, 1978; Doyle, 1979).

5. The manager can postpone evaluating an antecedent to work on more important problems.

While this approach provides greater control of the application of inference and knowledge acquisition methods in filling in missing information required by reasoning, it incurs two types of expenses. The first is the added cost of rule meta-evaluation. For each function in the antecedent, the corresponding schema must be accessed and interpreted according to its type: low-cost, variable, or multi. Hence, there is a per function constant overhead. In the KS system, the cost of interpretation has been removed. Each functional schema is both a declarative and procedural representation that can be executed directly without interpretive overhead (Fox, 1979a; 1981a). Hence, meta-evaluation costs are negligable.

The second expense is the increased complexity of the manager's rules to handle task selection, employee selection and resource assignment. Research in perceptual tasks, such as speech vision (e.g., HEARSAY-II), has shown the need for such capabilities when dealing with complex and uncertain data.

6. Acknowledgements

I would like to thank Herb Simon, John McDermott and Jaime Carbonell for their criticisms of earlier versions of this paper.

7. References

Davis, R., and King, J. 1975. *An overview of production systems*, Technical Report, STAN-CS-75-524, Memo AIM-271. Computer Science Dept., Stanford University.

Doyle, J. 1979. A truth maintenance system. *Artificial Intelligence* 12(3).

Erman, L.D.; Hayes-Roth, F.; Lesser, V. R.; and Reddy, D. R. 1980. The HEARSAY-II speech-understanding system: Integrating knowledge to resolve uncertainty, *Computing Surveys* 12:214-253.

Fox, M.S. 1978. Knowledge structuring: An overview. *Proc. of the Second Conf. of the Canadian Society for Computational Studies of Intelligence*, pp. 146-155. Toronto, Ontario.

[2]The problem of *attention focusing* (Hayes-Roth & Lesser, 1977) relies on good heuristics for choosing among alternatives, and a language for describing knowledge source capabilities (Fox, 1979b).

Fox, M.S. 1979a. On inheritance in knowledge representation. *Proc. of the Sixth International Joint Conf. on Artificial Intelligence.* Tokyo, Japan.

Fox, M.S. 1979b. *Organization structuring: Designing large, complex software.* Technical Report, CMU-CS-79-155, Computer Science Dept., Carnegie-Mellon University.

Fox, M.S. 1981a. *Knowledge structuring: Knowledge accomodation and discovery using specialization and planning.* Ph.D. thesis, Computer Science Dept., Carnegie-Mellon University.

Fox, M.S. 1981b. *SRL: Schema representation language.* Technical Report, The Robotics Inst., Carnegie-Mellon University, (in preparation).

Hayes-Roth, F. 1977. The role of partial and best matches in knowledge systems. In Waterman and Hayes-Roth (Eds.), *Pattern-directed inference systems.* NY: Academic Press.

Hayes-Roth, F., and Lesser, V. 1977. Focus of attention in the HEARSAY-II speech understanding system. *Fifth International Joint Conf. on Artificial Intelligence,* Aug. 1977, at Cambridge, MA.

Joshi, Aravind K. 1978. Some extensions of a system for inference on partial information. In D. Waterman and F. Hayes-Roth (Eds.), *Pattern-directed inference systems.* NY: Academic Press.

Klahr, Philip. 1978. Planning techniques for rule selection in deductive question-answering. In D. Waterman and F. hayes-Roth (Eds.), *Pattern-directed inference systems.* NY: Academic Press.

Lenat, D. B. 1976. *AM: An artificial intelligence approach to discovery in mathematics as heuristic search.* Ph.D. thesis, Computer Science Dept., Stanford University.

Lenat, D. B.; Hayes-Roth, F; and Klahr, P. 1979. Cognitive economy in artificial intelligence systems. *Proc. of the Sixth Annual Conf. on Artificial Intelligence,* pp. 531-6. Tokyo, Japan.

London, P. E. 1978. *Dependency networks as a representation for modelling in general problem solvers.* Ph.D. thesis and Technical Report TR-698, NSG-7253, University of Maryland, Sept. 1978.

McDermott, J. 1979. *ANA: An assimilating and accomodating production system.* Technical Report, Computer Science Dept., Carnegie-Mellon University.

Mostow, D. J. 1977. A revised architecture for rule-based systems. *Sigart Newsletter* 63:85-6.

Newell, A. and H. A. Simon. 1972. *Human problem solving,* Englewood Cliffs, NJ: Prentice-Hall.

Rosenberg, S. 1979. Reasoning in incomplete domains. *Proceedings of the Sixth International Joint Conf. on Artificial Intelligence,* pp. 735-37. Tokyo, Japan.

Shortliffe, E. H. 1976. *Computer-based medical consultations: MYCIN,* New York: American Elsevier.