A COMPARATIVE STUDY OF CONTROL STRATEGIES FOR EXPERT SYSTEMS: AGE IMPLEMENTATION OF THREE VARIATIONS OF PUFF

Nelleke Aiello
Heuristic Programming Project
Department of Computer Science
Stanford University
Stanford, California 94305

Abstract

This paper presents the results of comparing three control strategies for expert systems: event driven, expectation driven, and goal driven. Three different versions of a pulmonary function analysis system (PUFF) were implemented, each with a different control strategy. The systems are described and compared for efficiency and naturalness. The advantages and disadvantages of each strategy are discussed. The reasons why one approach, the expectation-drive strategy, is best suited for the PUFF application are summarized.

1. Introduction

Three versions of the PUFF pulmonary function analysis system are compared here, each using the same domain knowledge (represented as production rules) but different control strategies. The purpose of this research is to analyze these control strategies and the resulting programs and to examine what knowledge the knowledge engineer has or needs when choosing a control strategy for a new expert system.

The three different control strategies are described briefly as follows. **Event driven** (also called data driven) consists of input data or earlier *events* that lead to the invocation of rule sets that generate further events. With a **Model driven** strategy, the system matches the current "state of the world" against disease models to generate *expectations*. Further actions are taken based on expected symptoms. **Backchaining** is a *goal-driven* strategy in which a goal rule invokes all the rules whose conclusions are referenced by the conditions of that goal rule. These rules in turn invoke relevant rules in a chain until rules that reference only the input data are reached. The three PUFF systems were developed using the AGE system for building expert systems. A user of AGE can define production rules about a particular domain, set up a basic structure for a solution space, and then experiment with different control strategies to find one that best fits the problem.

There are several dimensions along which the *best fit* can be determined, both subjective and objective. Subjective measurements include an indication of how *natural* the knowledge represented and the output of the program seem to the expert. Objective measurements are speed and accuracy in terms of the total number of rules evaluated or comparisons with the physician's diagnosis.

1.1. PUFF Tasks

The basic task of PUFF is the interpretation of pulmonary function tests. These standard tests include measuring lung volume, that is, the volume of air and the rate at which it can be forcibly exhaled and the capacity to inspire a large volume of air. These and other laboratory measurements, as well as data about the patient's history and referral diagnoses are interpreted by production rules. The result is a set of interpretations and conclusions and a pulmonary function diagnosis, similar to those a doctor would produce, given the same initial data. The original PUFF system (Kunz et al., 1978) was a rule-based system that was designed with the EMYCIN system (van Melle, 1980) for building expert consultation systems. CENTAUR (Aikens, 1980) is yet another version of PUFF, using frames to represent prototypes of particular diseases and subtypes. The three versions of PUFF compared for this experiment use the same domain rules as the original PUFF system.

1.2. AGE

AGE (Nii et al., 1979), which stands for Attempt to GEneralize, is a collection of tools and partial frameworks for building expert systems. In the examples presented in this paper, we have exploited several AGE features. We were able to experiment with various designs and control strategies using the same domain knowledge. For the purposes of this experiment, we tried to use the same rules, data structures, and input data as far as possible for each system. Only the control strategies were varied. Initially, we implemented PUFF as a *blackboard model* (Erman et al., 1975) with a very simple, event-driven control strategy. It was made into an expectation driven system by adding a few rules for the initial diagnosis model and converting other rules to generate expectations for pulmonary diseases and their typical symptoms. Finally, we added a goal rule and one new attribute in the data structure to allow for the *backward chaining* of the domain rules.

2. Description of the Programs

2.1. Event-driven

The event-driven version of PUFF, AGEPUFF, has a simple blackboard data structure to store the intermediate and final results. There are two levels, the PATIENT level and the DISEASE level, with three instance nodes on the DISEASE level: OAD (Obstructive Airway Disease), RLD (Restrictive Lung Disease), and NORMAL. An initial set of rules (termed a *knowledge source* in AGE) looks at all the data (the test measurements) and records what those measurements indicate on the blackboard. Each action of recording something on the blackboard is called an *event*. When all of the relevant rules in the initial knowledge source (KS) have been evaluated, a user-specified *selection method* chooses one of those events to use as a *focus*. This focused event is then matched against the *preconditions* of all the knowledge sources, and the one whose precondition matches the focused event is invoked next. As an example (see Fig. 2-1), events that indicate some conclusion about the degree of OAD trigger a KS (OAD-SUBTYPERULES) that tries to determine the OAD subtype (Emphysema, Asthma, or

Bronchitis). In other words, the events are driving the order in which KSs are evaluated. This process is repeated until some termination condition is met.

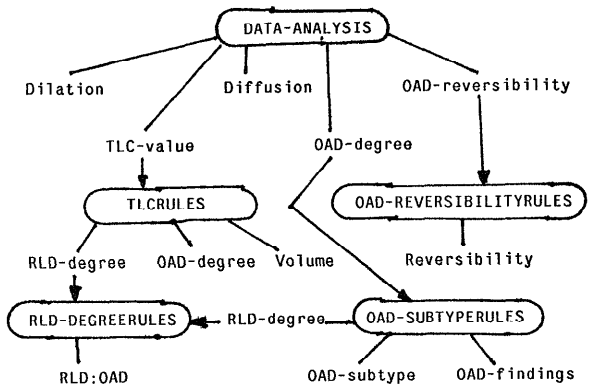

Figure 2-1: AGEPUFF knowledge sources
(with circled nodes indicating knowledge sources).

The following is an example of a rule from the event-driven PUFF system.

```
if (BETWEEN* ($DATA SLOPE:F50/FVC) 22 32)
then PROPOSE ev.type OAD-DEGREE ch.type
        MODIFY  hypo-element 'OAD
        attr-value (DEGREE '(MODERATE .5))
        (FINDINGS '((F5025 *F5025* ) 1.0]
```

If the condition of this rule is true, the DEGREE and FINDINGS attributes of OAD will be updated on the blackboard, generating an event of type OAD-DEGREE. This event is added to a list of events and becomes the focus event on a last-in, first-out basis. A knowledge source with a precondition of OAD-DEGREE can then be invoked, and OAD would become the focus node.

2.2. Model Driven

AGEPUFF/MODEL is the name of the model-driven version of PUFF implemented in AGE. It uses the expectation control strategy available in AGE to set up models of the pulmonary diseases. The initial diagnosis rules check a limited number of crucial data items and make an initial, broad diagnosis. AGEPUFF/MODEL then attempts to substantiate the initial hypothesis with a few more data items of secondary importance. If the diagnosis is still credible, a set of further model-based expectations is generated for corroborating evidence. AGE compares the expectations with input data and partial hypotheses on the blackboard. Figure 2-2 shows the organization of the knowledge sources for the expectation-driven model.

The models of OAD and RLD help to focus the interpretation of the input data. For example, if a patient has a total lung capacity greater than 120 percent of the normalized value for his size and age, it is very likely that he has OAD. So, instead of checking all the available test data, as in the event-driven example, AGEPUFF/MODEL looks first, and possibly only, at other indicators of OAD. If enough other indicators (test data, calculations, patient

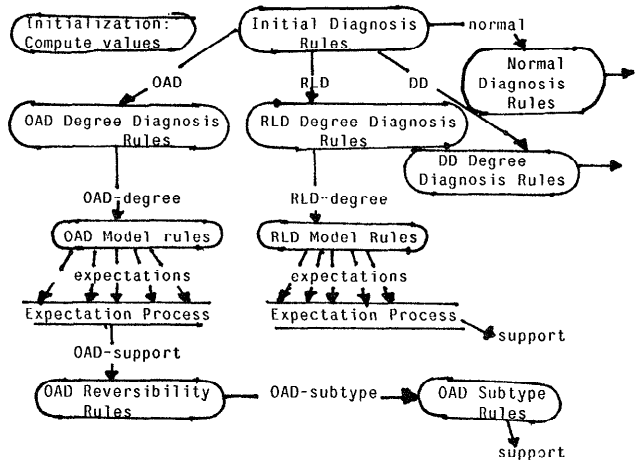

Figure 2-2: Expectation driven organization of AGEPUFF/MODEL.

history, and referral diagnosis) agree with the OAD diagnosis, then the actions that reinforce the initial conclusion are taken; which in turn generate events that trigger knowledge sources about OAD reversibility and OAD subtypes.

The following is an example of a rule from the model-driven system.

```
if (EQDEG* ($VALUE 'OAD DEGREE) MODERATE)
then EXPECT expr (BETWEEN* ($DATA
                SLOPE:F50-F25/FVC) 22 32)
             actions (PROPOSE ev.type OAD-SUPPORT
                ch.type MODIFY hypo-element 'OAD
                attr-value ((DEGREE (MODERATE .5))
                (FINDINGS '((F5025 *F5025*) 1.0]
     EXPECT expr (GREATERP* ($DATA FVC/FVC-PRED)
                80) (BETWEEN* (DIFFERENCE ($DATA
                FEV1/FVC) ($DATA PRED:FEV1/FVC))
                -25 -15)
             actions ...
     EXPECT ...
```

When this rule is executed, several expectations are generated. The expectation expressions (expr) are evaluated in a last-in, first-out order and, if true, the actions are taken. These actions can generate events, just as in the event-driven system.

As noted above, the expectation-driven approach is really a combination of an event and expectation strategy. No system can function on expectations alone. Events are used to generate information for the expectations and to act when an expectation is met.

2.3. Backchaining

The backchained version of PUFF, called PUFF/BC, has only one knowledge source. All the PUFF rules are combined into one large knowledge source. The rules are *chained* by looking at the conditions of the *goal rule* and then searching for other rules whose actions might provide values for the goal rule's conditions. This imposes a more restrictive syntax on the conditions of rules than in the event-driven or expectation-driven approaches. The backward chaining process requires that the conditions be written as predicates, with explicit object and attribute names. (In the other two systems, the predicates can be arbitrary LISP expressions.) AGE automatically chains the rules by searching for rules whose actions reference the same object-attribute pairs as the conditions

of the goal rule. This chaining process is repeated, with the rule whose actions referred to the desired object and attribute becoming the next subgoal rule, until conditions are rxeached that only reference input data.

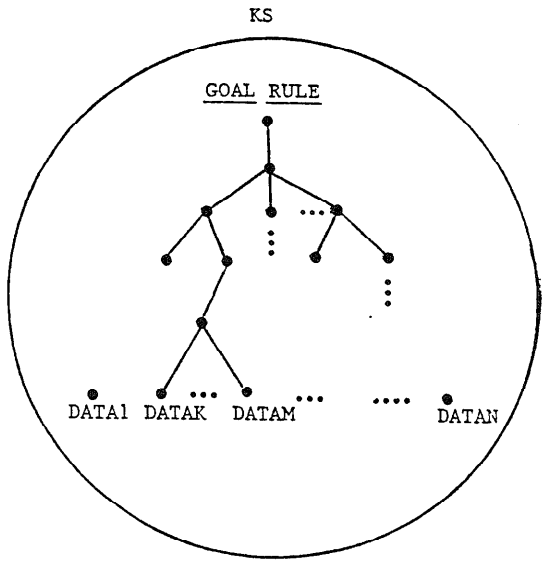

Figure 2-3: Organization of knowledge sources for Backchaining.

An example of a rule from the PUFF/BC system follows.

```
if (BETWEEN* DATA SLOPE:F50-F25/FVC 22 32)
then object OAD attribute-value pairs (DEGREE
     ('MODERATE .5))(FINDINGS ((F5025 *F5025* 1.0]
```

In this case, the value of the attribute SLOPE:F50-F25/FVC of the node DATA will be passed on as an argument to the predicate BETWEEN*. When this rule is evaluated, rules referring to DEGREE and FINDINGS of OAD in their predicates can be evaluated next.

3. Results of Comparing the Three Strategies

3.1. Speed

The three control strategies were objectively compared for efficiency by measuring the number of rules tested, the number of rules executed, the number of input data items referenced, and the number of references to the blackboard or internal data representation in the backchained strategy. Table 3-1 shows the average results for the three strategies for a small sample of actual cases with diagnoses that range from mild to severe cases of disease categories OAD, RLD, and NORMAL, and various combinations thereof. There was very little deviation from the average for each measurement.

From this objective measurement it is clear that the model-driven strategy does less testing and refers less often to data than the other two strategies do. The ability to focus initially on the most likely diagnoses eliminates the need to test rules for other diagnoses. A few more rules are executed in the model-driven and goal-driven strategies to set up the expectations or goals. The goal-driven strategy has the most references to internal data because every predicate refers to an object-attribute pair, each rule may have several predicates, and all the rules are tested for each run.

3.2. Accuracy

Accuracy was measured as agreement with the doctor's conclusions, based on the statements PUFF is capable of producing. When the strategies are compared by the number of correct interpretations and diagnoses, the event-driven and goal-driven strategies are slightly more accurate than the model-driven strategy. Given odd or marginal data, the model-driven strategy may produce incomplete interpretations. It will not produce incorrect statements, however. The event-driven strategy is initially data driven and therefore responds to each data item, odd or normal. The goal-driven strategy evaluates all of its rules, independent of the particular input data. In other words, the event-driven and goal-driven strategies will produce all conclusions derivable for a given set of input data. But the model-driven strategy will produce only conclusions compatible with its initial diagnoses and wil produce results derivable from the models for those diseases.

3.3. Naturalness to the Expert

Accuracy can also be evaluated subjectively in terms of the naturalness of the knowledge represented and the output. A knowledge representation that seems natural to the expert facilitates knowledge acquisition from the expert and comprehension of the system by the expert. A natural output includes the order in which interpretations are generated: They should be produced in an order similar to that produced by the expert.

3.3.1. Output

The order of the output of interpretations in all three PUFF systems is determined by the order in which those interpretations are concluded. We compared these orders with the interpretations from the original physician's reports. For each case, the number of items out of order was calculated by counting the minimum number of *moves* required to reproduce the doctor's order. Output generated by the model-driven system always had fewer findings out of order. The event-driven system had the next fewest out-of-order items, and the goal-driven system had the most. This result was not unexpected, since the model used by AGEPUFF/MODEL is based on the doctor's order of reasoning from pulmonary test measurements to pulmonary disease diagnoses. The backchained system produced the most items out of order because the order of execution of the rules is determined by the chaining of attributes. A rule is executed because another higher goal rule needs the value of one of its concluding attributes, but all other conclusions of that rule will be made at the same time, even though those conclusions are not required until later in the processing.

3.3.2. Knowledge Representation

The domain knowledge used by each of the three control strategies is basically the same. All three use production rules to represent the domain knowledge. They differ in how those rules are

Table 3-1: Measured Comparison of Three Control Strategies

Strategy	rules tested	rules executed	input data	internal data
event-driven	60.4	12.6	80.4	54.4
model-driven	35.5	14.5	47	48.5
goal-driven	68	14	76	128.8

organized. In the event-driven approach, the rules are divided into sets called knowledge sources. A knowledge source is invoked when its precondition matches the event type of the focused event. The model-driven system also has sets of production rules, but some of those rules create expectations instead of generating events. The expectations (in this case, a set of ranges for data values to support the focused initial diagnosis) are matched against input data and the partial solution stored on the blackboard, before further KSs are invoked. If an expectation becomes true, it becomes the new focus. In the backward-chained system, the rules are not stored in any particular order or grouping. The order of evaluation and the focus are determined at run time by the backward chain from the goal rule.

3.4. Advantages and Disadvantages of Each Strategy for Implementing PUFF

The event-driven control strategy is robust; it considers all of the input data, and tries to follow through on every event generated. Because of this thoroughness, it worked correctly even in the presence of unusual and incomplete data. However, the event-driven approach can be nonconvergent. Event-driven systems can only produce conclusions derivable directly or indirectly from the input data. They cannot focus on or direct the search toward a desired solution.

The expectation-driven model is usually more efficient than either of the other two examples. It can focus on important factors and come immediately to the correct conclusions for most cases. However, odd or conflicting data, that do not fit the model will cause problems. Thus, a disadvantage of the model-driven paradigm is that the correct solution depends heavily on the *correct* model and initial focus. If the system begins by focusing on an incorrect diagnosis, it will check only the data that are relevant to that wrong diagnosis. Data indicative of another diagnosis would be ignored. The importance of the correct model was demonstrated during the implementation of AGEPUFF/MODEL. The first model implemented worked correctly for extreme indications of OAD and RLD but expected everything *in between* to be normal. The normal expectations were correctly unconfirmed, but the system could not go back and correctly diagnose mild or moderate OAD or RLD.

The backchain example has several advantages. First of all, it is simple for the user to understand and it lends itself to easy explanation of its reasoning. At each step of the execution, the next step is predetermined. The rules will always be evaluated in the same order, regardless of the particular input data. However, this does make the backchaining approach inefficient. Many of the rules in the backward chain may be totally irrelevant for a particular set of input data values, especially normal, healthy values. It cannot focus on particular data items in different sets of input data because it has no mechanism for deciding what is important and what is not. The goal of diagnosing pulmonary diseases is the same for every case, for every set of test data.

4. Conclusion

All three approaches use a divide-and-conquer approach to solving the PUFF problem, but they differ in how the domain knowledge is divided. In the event-driven system, knowledge is divided by its association with the input data. A particular value of an input data item can cause a rule to conclude the presence or severity of a disease. That disease then becomes the focus for the next set of rules (KS) evaluated. Or several input data items can together cause a rule to conclude a new value for some item on the blackboard that then becomes the new focus. The focus of the system stays with a particular input data item, and all conclusions based on it, until there are no more KSs to draw further conclusions.

The expectation-driven system organizes its knowledge around its disease models, the expected symptoms associated with each disease. The focus of the system varies with the initially hypothesized diagnosis. This focus is much stronger than that of the event-driven system. In most cases, the system considers only one or two diseases, making all possible conclusions about one disease before considering the alternatives. In the event-driven approach, each individual input data item could trigger a short focus on a particular diagnosis, with support building with each additional focus on the same diagnosis.

The goal-driven strategy is focused by the backward chain connecting the production rules. It looks at data relevant to the goal rule's conditions and thus focuses on each disease, present or not. In the event-driven and goal-driven approaches, all of the input data have the same importance. In the model-driven approach, some data items are more important than others. Input data used by the model to generate initial diagnoses are always referenced; data items referring to diagnoses not in the initial hypothesis may be ignored.

The characteristics of the PUFF problem that make it a good application for an expectation-driven control strategy include:

1. A large amount of input data.
2. A small solution space (three possible diseases plus a small number of subtypes).
3. A simple model for initial hypotheses.

Applications with similar characteristics are signal processing and other diagnostic problems. Other systems written in AGE using the expectation-driven control strategy include AGE-VM, a ventilator management system (Fagan, 1980), and GEO, a geological data interpretation program. Each new application is likely to have slightly different characteristics from the last, so it is not reasonable to expect one control strategy to satisfy everyone's needs.

5. Acknowledgments

This research was supported by the Advanced Research Projects Agency under Contract No. MDA 903-77-C-0322 and the National Institutes of Health Grant No. RR-00758.

6. References

1. Aikins, J. S. Prototypes and Production Rules: A Knowledge Representation for Computer Consultations. Doctoral dissertation, Heuristic Programming Project, Dept. of Computer Science, Stanford University, 1980.

2. Erman, L. D., and Lesser, V. R. "A multi-level organization for problem-solving using many diverse cooperating sources of knowledge." In Proc. IJCAI-75, pp. 483-490.

3. Fagan, L. M. VM: Representing Time-Dependent Relations in a Medical Setting. Doctoral dissertation, Heuristic Programming Project, Dept. of Computer Science, Stanford University, 1980.

4. Kunz, J., Fallet, R., McClung, D., Osborn, J., Votteri, B., Nii, H. P., Aikins, J. S., Fagan, L., and Feigenbaum, E. "A Physiological Rule Based System for Interpreting Pulmonary Function Test Results." HPP-78-19 (Working Paper), Heuristic Programming Project, Dept. of Computer Science, Stanford University, December 1978.

5. Nii, H. P., and Aiello, N. "AGE: A knowledge-based program for building knowledge-based programs". In Proc. of IJCAI-79, Tokyo, Japan, August, 1979, pp. 645-655.

6. Van Melle, W. *EMYCIN: A Domain-independent Production-rule System for Consultation Programs*, Doctoral dissertation, Heuristic Programming Project, Dept. of Computer Science, Stanford University, 1980.