BUILDING EXPERT SYSTEMS FOR CONTROLLING COMPLEX PROGRAMS

Sholom Weiss, Casimir Kulikowski, Chidanand Apté, Michael Uschold
Department of Computer Science, Rutgers University

Jay Patchett, Robert Brigham, Belynda Spitzer
Amoco Production Research

Abstract

Production rule schemes have proven quite effective in concisely representing expert knowledge in several application areas. Yet, there are many problems for which one would like to take advantage of additional knowledge that may not be easily represented in these surface level models. One class of problems of particular practical interest are those in which we would like to have a computer-based system give interactive advice on how to control and interpret results from a set of complex and interrelated applications programs. The advice may refer to interpretations of current results, possible experiments that should be performed with the help of the applications programs, and indications of inconsistencies in specific analytical procedures and in problem solving sequences followed by the user. In the present paper we report on our experiences in designing an expert system (ELAS), of the type described above, for well log analysis in oil exploration and production. We have integrated a production rule advice model (using the EXPERT system) with existing Amoco software for well-log analysis and display. In doing so, the original system for well-log analysis was reorganized so that its use could be monitored and controlled, and its knowledge structured according to the types and sequences of methods used by expert analysts. By varying the assumptions and parameters used in the different individual analyses, our goal is to make available interactive interpretations of the alternative approaches that an expert might take to a complex problem of well-log analysis.

I Introduction

In a recent article [1], Hart describes several of the research issues arising in the design of multi-level expert systems. He contrasts *surface level models* with *deep models* of reasoning. By way of example, Hart describes a hypothetical system for advising petroleum engineers using a multi-level approach. The surface level model is of the production rule type, whereas the deep model is a purely mathematical description of an oil reservoir expressed as a set of partial differential equations. The latter is typically implemented as complex software tools, such as reservoir simulators.

We have built a multi-level expert system called ELAS (Expert Log Analysis System) for carrying out well-log analysis. Well logs are the various electromagnetic, sonic and nuclear signals obtained from instruments placed down-hole in a well, which characterize the properties of the rock and fluid formations around the borehole. From a practical applications point of view, well log interpretation represents an important problem, since it permits an assessment of the likely presence of hydrocarbons and possible yields of the well during exploration and production. From the perspective of expert systems research, this application is proving very helpful in increasing our understanding of representation, communication and control processes in multi-level systems. And, from the more general software engineering point-of-view, we are learning how one might exploit existing software systems more fully by building a coordinating and advisory system that makes these programs easier to use by a wider variety of expert and non-expert users alike.

In many problem areas, it is not unusual to find that valuable software has already been developed to aid the expert in data analysis, the design of experiments, and the interpretation of results. These programs are often quite complex packages, developed over several years and enhanced through extensive user experience. In designing an expert system, it is only natural that one should want to take advantage of such software. One of the first efforts in modeling expert advice on the use of a complex program was the SACON [2] project which developed an advisory model for the MARC structural analysis program. However, there was no interaction between the two programs: SACON was run before the MARC program, giving advice on its prospective use. In order to develop an expert system to its fullest potential, interaction is needed between the advising program and the application programs. In a sophisticated system, the interpretive program will be fully integrated with the application programs, so that they communicate their results to one another, and advice changes dynamically as the model tracks the user interaction. Furthermore, the system must have the ability to automatically take a recommended action if the the user agrees. In effect, we will have a program that not only gives advice, but also can accept the advice and act on it.

II Comparison with a Interpretation/Classification Model

An expert model which interacts with a complex program can be thought of as an extension of the type of interpretation/classification models that have been widely used in medical consultation (such as CASNET and MYCIN), and geology (PROSPECTOR). Regardless of the underlying knowledge representation, these models are typically implemented as programs which ask questions of the user until enough evidence has been accumulated for the model to offer an interpretation. In situations where the human experts follow agreed upon procedures for eliciting evidence, the questions are often highly structured, and one can expect that the results will be reported in a systematic and ordered fashion. Although some of the results may be gathered by external sources (such as instruments), they are typically "filtered" through the user who enters them into the program. In a few situations, where most of the evidence is taken directly from instrument data, [3, 4], signal processing algorithms feed directly into the reasoning model.

In a model that is completely integrated with a set of

complex applications programs, so that they appear to the user as a single program, important additional functional characteristics are needed that go beyond those found in the typical interpretation/classification type models:

- The advisory model must be able to receive data or evidence automatically from the application programs, in addition to that reported by the user.

- The logic of the model must be prepared to interpret evidence in real-time.

- The model must be able to not only suggest advice on the interpretation of evidence, but must also monitor how the user reacts to this advice in his subsequent choice of methods of analysis, and then provide new advice (to fit the dynamically changing situation) if the user requests it.

Yet, despite the above differences we can use an important analogy from the more traditional consultation systems: the set of applications programs can be made to communicate with the advice model primarily through the evidence, as long as we are willing to broaden our definition of what we consider to be evidence. We will now take evidence to include not only facts about the particular domain problem, but also the status of the actions taken by the user while interacting with the applications programs, and the results of key calculations performed by these programs as the result of the user's actions. In the advisory model, we must represent classes and sequences of user actions, their consistency relationships, and expected effects. To be both effective yet flexible to the nuances of expert problem solving, the advisory system must monitor and interpret the actions of the user in a type of *background mode*. The user must have a considerable degree of freedom in the choice of methods and direction of analysis, yet should be discretely warned if his choice is leading to a likely dead-end, or is introducing harmful inconsistencies in the analysis or interpretation. If the analysis is proceeding smoothly, as might be expected with an experienced user, the model should be available in an advisory and summarizing capacity, without interfering with the user's control over the problem solving flow.

The logic for the interpretive analysis can be stated in the form of production rules as in the usual interpretation/classification models. However, the designer of the model must take into account the newly enlarged scope of types and the dynamically varying nature of the evidence that must be handled. While in a traditional consultation system we expect evidence to remain relatively stable during a single consultation session, here, because the control and monitoring of the separate applications programs must be carried out in real time, evidence will change values very frequently in the course of a session. For example, an observation such as, *Task A not performed*, will be changed once Task A is done; or a numerical result may be received from one program, and then be changed as the consequence of further analysis and processing by another program. This type of situation resembles a consultation where someone is continually modifying or updating the evidence, and sequentially asking for an interpretation. There are some analogies to the medical problem of evaluating a patient over time. Here, however, the time frame is highly condensed into a real-time evaluation.

Another important difference with the usual interpretation/classification model is that the controlling model has the opportunity to go beyond just giving advice; it can provide the means of accepting the advice and taking the recommended action in real time. This step may prove somewhat more difficult for expert system builders than might be initially expected, since the model and the set of application programs must be unified in what appears to the user as a single system. In addition, a single task of the expert may require the compilation of many steps through one or several of the programs or even combining or adding steps that are not directly available in the application programs. Although the application programs may not have to be completely recoded, new code may be needed to integrate steps that reflect the expert's particular approach to analyzing a problem. In our experience, we have found that this can be done successfully while building an overall expert system.

III The ELAS Expert System for Well-Log Analysis

The main goal of the ELAS project is to introduce methods of expert systems in well-log interpretation. In ELAS we are demonstrating how the knowledge and reasoning methods of an expert log analyst can be combined with INLAN - Amoco's large scale interactive program for well log data analysis and display [5]. The Dipmeter Advisor, which is a knowledge-based interpretation model for a very specialized type of log analysis, has been described previously by others [6]. In our system, we are concentrating on the more widely used and more numerous suites of logs for which there is quite a large body of published [5] and unpublished results .

Working with the expert analyst, we have observed the sequence of steps involved in solving certain typical interpretation problems, and encoded them as a set of advice rules using the EXPERT [7] formalism. Thus, more specific objectives in developing ELAS include:

- formalizing methods of expert well-log analysis into a representation that can be easily used and understood by others;

- providing interpretations based on the data, the actions, and expectations of the user and the program

Our first prototype system is running on IBM's VM/CMS operating system using a RPQ terminal configuration. This configuration consists of a Tektronix graphics terminal with a joystick, an IBM CRT, and a single alphanumeric keyboard. The system allows the user to interactively perform experiments in the analysis of logs. Advice is generated based on the results of previous experiments, and a running summary is kept of the actions already taken. The advice system is fully integrated with INLAN, while based on the EXPERT production rule scheme.

One of the contributions to a specific domain that comes from building an expert system is that it helps to structure and organize practical problem solving knowledge

in the domain. When many of the important methods of interpretation are informally specified, and personal to the experts in the domain, the expert systems formalization can be particularly valuable. By encouraging or even forcing a formal structure on the domain knowledge, and incorporating it into a runnable expert system, the knowledge becomes testable, reproducible and more widely reportable to others. While many general software programs require expertise not only in the application domain but also in the use of the program, our objective in developing ELAS has been to produce a new, powerful yet easy to use experimental tool for the well-log analyst.

To make interaction easy for users, the front-end of the ELAS system has as its top level a *Master Panel*, which holds a snapshot of the current status of the analysis of an already selected well (Figure III-1).

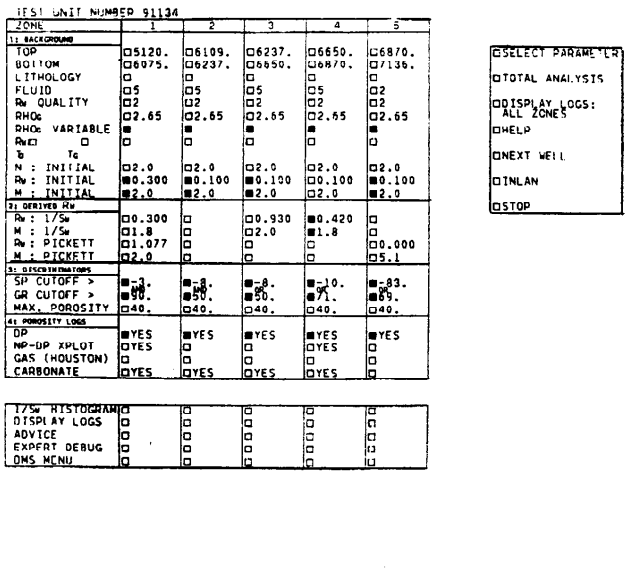

Figure III-1: Preliminary Master Panel

Most user-program communication is controlled through this master panel. It is displayed on the Tektronix graphics screen, and includes a concise set of key parameters and tasks that are crucial in well log analysis. Here, a parameter may be a constant, a log (represented as a vector of digitized values for each foot of depth in the well), or an expected characteristic of the well (e.g. the presence of gas in some zone). The value of the parameter will influence the results of the subsequent analysis. Whereas the original software was an interactive system that required specification of many detailed operations, our task has been to develop a much higher level system that compiles many of these smaller steps. At the initial user level there is a superficial similarity to VisiCalc, the highly successful personal computer program. In the simpler environment of VisiCalc, we see a program that presents information in a concise format and allows the user to vary a parameter and then watch all dependent results change. In our case, we are faced with a much more complicated computational task, but we too try to show the propagation of effects that follow from the user's change of a parameter value or choice of analysis method within as short a time as possible, ranging from almost instantaneous to many seconds. This is done by updating the master panel at the top level, from which the user will be led to more detailed panels or displays for the specific methods. Changing a parameter may imply quite a large number of computational steps and not all information can be described in a simple tabular format.

In ELAS the user can direct both the mathematical analysis and the interpretive analysis by changing parameters or invoking tasks through the master panel. As just stated, the outcomes of mathematical analyses that follow are then reported back to the user through this same panel. The expert system keeps updating its interpretive analysis after every change in the evidence so that it always reflects the current status of the panel. The system also synchronizes all derived logs which are affected by changes in the panel parameters or methods. Changes are made either through user action or updates in the mathematical analysis. The user has the freedom to carry out an entire well log analysis sequence without ever asking for advice from the system, or he can proceed to request advice for any stage where he feels the need or curiosity for it.

The example in Figure III-1 shows the master panel for a sample test well, TEST UNIT 91134. The five columns correspond to five different geological zones (by depth) which have been chosen for analysis. The rows correspond to the parameters for the zones. Initial values for some of the parameters must be supplied by the user. Many of them, however, may be obtained through subsequent analysis. Some parameters may stand for specific tasks that the user might want to invoke to help in the analysis.

Consider another sample test well, NO. 25 TEST FIELD UNIT, for which the user wants to examine (on his own, or following the advice of the model) the effect of lithology on the calculation of rock porosity in a zone. The appropriate box on the panel for that method and zone is marked. The marking of that box causes the system to invoke the task, displaying first a crossplot of the zone depth points with the various lithological choices made available for further analysis (Figure III-2). These are shown as the SS (sandstone), LMST (limestone), and DOL (dolomite) boxes on the crossplot. Once the user has made his choices (of two of these boxes, in this particular instance), the program calculates the corrected logs using the lithological assumptions, and then returns to the main panel, which now reflects any updates due to this task invocation. This is an example of a single step in an incremental analysis. If the user wishes to see the full extent of analyses that follow a particular change in parameter value, all he needs to do is mark the TOTAL ANALYSIS box on the side of the master panel, and a set of derived logs, which are particularly conclusive, will be displayed. In addition, the interpretation for each geological zone will be given. The logs displayed in Figure III-3 were plotted using a batch plotter program. They represent the type of display we are designing, although the current version of ELAS does not shade the overlaid portions of logs while displaying them.

The system also maintains consistency between dependent tasks, which is necessary whenever a significant parameter changes values. This can involve computations

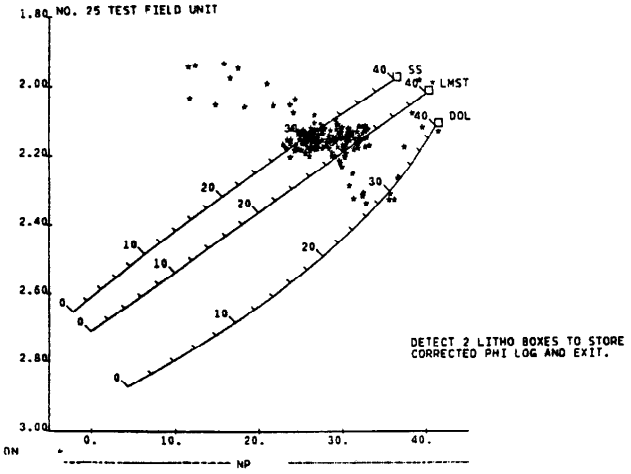

Figure III-2: Neutron Density Crossplot

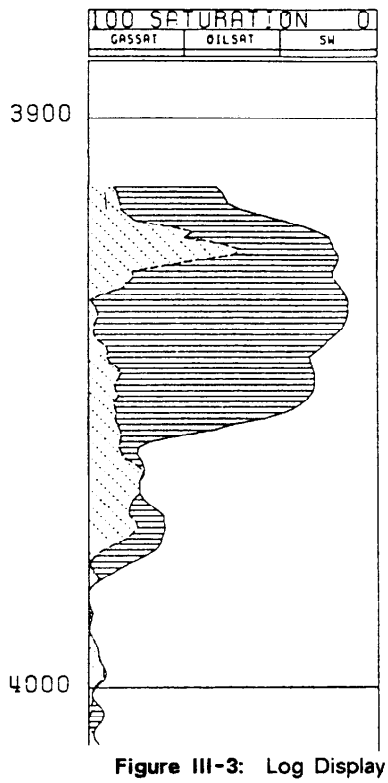

Figure III-3: Log Display

ranging from almost negligible to very formidable. For example, one of the most frequently used formulas in log analysis is:

$$S_w = (F \cdot R_w / R_t)^{.5} \qquad (1)$$

where the variables are quantities that can be changed through the master panel. Even a minor change in one of the variables in this equation involves a recomputation for all the points in the log, which are usually in the thousands. This will immediately cause reinterpretation and revision of previous conclusions and recommendations.

In summary, we are emphasizing that the formalization of the methods may very well require a new organization and presentation of the original software, and may not be quite as simple as feeding numbers and arguments back and forth between the original software and the model. A clean interface is of course an important ingredient in the ultimate success of the program. In our case, the system will lead the user into many specialized routines associated with a particular task. Often, the setting of a key parameter may require a separate analysis in itself.

The well log analysis model considers 3 types of advice which can be described in a production rule framework.

1. interpretation of existing evidence and past actions
2. advice on future actions
3. consistency checking

In a system such as ELAS, there are many possible reasoning paths that the user may take, most of which are under user control. Thus depending on the user's prior actions, the model may give radically different advice even for the same initial data. Furthermore, depending on its interpretation of the status of analysis and user actions up to a given stage, the model will make recommendations to try out various preferred methods for subsequent analysis. The user also has the choice to enter certain *a-priori* information about the problem, such as whether one ought to expect gas in the well. In this case, if we were told not to expect gas, but gas is indicated by an analysis of some of the logs, we can proceed to get clues as to whether the method of analysis might be at fault, whether the logs are noisy or otherwise inaccurate, or whether some underlying assumption is unjustified, etc. This is an example of the kind of consistency checking that involves both the log data and the interplay between the various methods of analysis. While actual advice depends on the types of analyses that are specific to a domain, we have found that they are all readily representable in a production rule framework.

IV Communication Between Programs

Because we are starting with existing software, we initially see two separate programs, the original software (INLAN) and the interpretation program (EXPERT). To integrate the two, we need a means of communication, which involves automatically filling in the arguments of some of the evidence necessary to correctly invoke the production rules. Figure IV-1 is a simple illustration of such a production rule, where the finding of gas can be communicated once certain tasks have been performed in the well log analysis program. This requires that the original program record such information and pass it on to the interpretation program. Because both programs are written in FORTRAN, communication is relatively straightforward.

If:
The neutron-density crossplot has been performed,
and gas is found,
and the current porosity log has not been gas corrected

Then: The following advice is given:
The porosity logs may be gas corrected by Methods a, b.

Figure IV-1: Example of a rule for advising on methods

Secondly, we need to communicate back to the well log program so that the user may now choose whether to accept the advice or not. The selection of methods through the master panel and other screens is menu oriented and therefore, we can use a dynamic scheme to place an item on the menu. A production rule may then be invoked to indicate the circumstances under which the method is displayed on the menu and is available for the user to select. Thus in the example of Figure IV-1 Methods *a* and *b* would appear on the menu when the production rule is satisfied. Figure IV-2 gives an overview of communication channels used in ELAS.

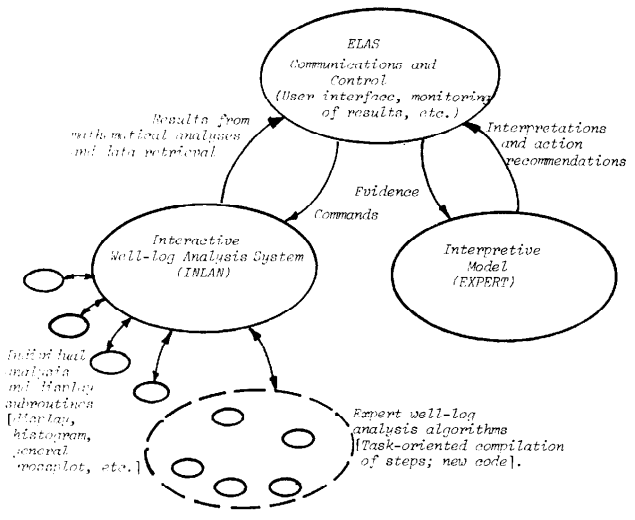

Figure IV-2: Overview of ELAS Communication

V Conclusions and Future Directions

A major challenge of this project is to develop an expert system that is effective and productive in a realistic, large scale application. In the present project there is a strong incentive to encode expert knowledge and make this knowledge available to others in the field because of the scarcity and cost of expert interpreters. The building of such a system will allow us to gain important insight into the complete spectrum of tasks needed for taking an expert system from conception and design to actual use in a real world setting. Another long-term goal of the ELAS project is to abstract some of the general principles of knowledge organization and systems design that are applicable to signal interpretation problems of which well log analysis is an example. Of particular interest to us is the development of more general schemes to interface an interpretive program to existing applications software. We have already established some of the mechanisms in this project, but a more structured approach may prove valuable in addition to a production rule scheme. Formal representation issues regarding multi-level expert systems are being presently studied within the framework of the ELAS system. Specification of inter-level communication, such as task dependencies, task-observation relations, and consistency invocations should eventually have schemes for easy encoding in the knowledge base. For example, task dependencies, such as *if Task A is redone then Task B must also be redone*, should be more concisely representable than in a pure production rule knowledge base.

The system is expected to be operational in Amoco's regional offices in the latter part of this year. If well-received, we expect that the system will grow in expertise, benefiting from the contributions of many knowledgeable users.

VI References

[1] Hart, P. "Directions for AI in the Eighties." *SIGART Newletter.* 79 (1982) 11-16.

[2] Bennett, James S. and Engelmore, Robert S. "SACON: A Knowledge-Based Consultant for Structural Analysis." In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence.* Tokyo, Japan, 1979, 47-49.

[3] Weiss, S., Kulikowski, C., Galen, R. "Developing Microprocessor Based Expert Models for Instrument Interpretation." In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence.* Vancouver, Canada, 1981, 853-855.

[4] Fagan, Lawrence M. "Ventilator Manager: A Program to Provide On-Line Consultative Advice in the Intensive Care Unit", Technical report HPP-78-16, Heuristic Programming Project, Stanford University, September 1978.

[5] Martner, Samuel T. and Brigham, Robert J. "An Interactive Computer System for Well Log Analysis." In *SPWLA Sixteenth Annual Logging Symposium.* , June 4-7, 1975, .

[6] Davis, R., Austin, H., Carlbom, I., Frawley, B., Pruchnik, P., Sneiderman, R., Gilreath., J. "The Dipmeter Advisor: Interpretation of Geologic Signals." In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence.* Vancouver, Canada, 1981, 846-849.

[7] Weiss, Sholom, and Kulikowski, Casimir "EXPERT: A System for Developing Consultation Models." In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence.* Tokyo, Japan, 1979, 942-947.