Application Design:
Issues in Expert System Architecture

Harry C. Reinstein
Janice S. Aikins

IBM Scientific Center
1530 Page Mill Road
P. O. Box 10500
Palo Alto, Ca. 94304 USA

ABSTRACT

We describe an expert system that has been applied to the task of application design. Users supply the system with problem specifications, such as the required output data, and the system produces a graphic representation of the completed application in the form of a flow diagram. The application design task has forced us to consider two important issues in expert system architecture: constraint processing and the explicit representation of control flow. The resulting knowledge representation and control logic are discussed.

I Introduction

Computers are commonly used in a wide variety of application design areas. Building the application programs, however, is often a difficult and frustrating task requiring the blending of both domain-specific and computer-specific expertise. It is frequently the case that the application problem expert is not a computer expert and must seek the help of appropriate computer professionals whose availability and skills may be limited. This problem has received considerable attention and work is proceeding along several paths in pursuit of an effective solution [1,2].

This paper presents an expert system which assists in preparing a well-formed definition of a desired application. This definition is presented in the form of a program flow chart showing the proper configuration of input devices, data processing functions, and output devices needed to produce the desired result. The flow chart is developed from a knowledge base which includes representations of data transformations valid within the problem domain. For example, if the problem domain is payroll applications a transformation relating GROSS_PAY to NET_PAY would be represented in the knowledge base and would include references to data processing functions required to enact the transformation.

A notable feature of application definition is the dual nature of the expertise required; an effective consultant must have knowledge of both the specific application domain (e.g., payroll applications) and of more general application design principles. Several interesting problems in expert system architecture have been encountered, especially in the areas of constraint processing and control flow. A discussion of these areas comprises the bulk of this paper. We begin, however, with an elaboration of the task performed and a discussion of our choice of knowledge representation and constraint architecture.

I The Task

Figure 1 shows a flow diagram of a simple payroll application. The task of the system is to produce this chart from an initial statement of the desired program outputs.
The system is expected to select appropriate devices and processing functions for the application and to infer or obtain values for relevant application parameters (e.g., file names). The driving mechanism for this process is a search to find a sequence of known data transformations (e.g., GROSS-PAY to NET-PAY) which begins with device-available data and ends with the desired output data. Transformation descriptions include knowledge of the processing functions required to effect the transformation. These functions may themselves introduce additional data elements into the flow diagram. Processing each element of the flow diagram until there are no unambiguous references to other elements will produce a complete application description.

The consultation dialog involves the user in two ways. First, there is the requirement to obtain non-inferrable parameters such as APPLICATION-NAME. Second, the system enlists the user in the selection of specific elements of the application flow diagram whenever the application definition is inadequately constrained to permit an unambiguous, automatic selection. This part of the consultation may be conducted at several levels, ranging from direct inquiry for the identity of the element to the acquisition of application constraints that may themselves implicitly permit the disambiguation of the element in question (3 I.

III The Knowledge Base

The primary knowledge in the system is organized into frame-like [4,5] objects with attributes. Each element of the flow diagram is represented as a particular frame in the knowledge

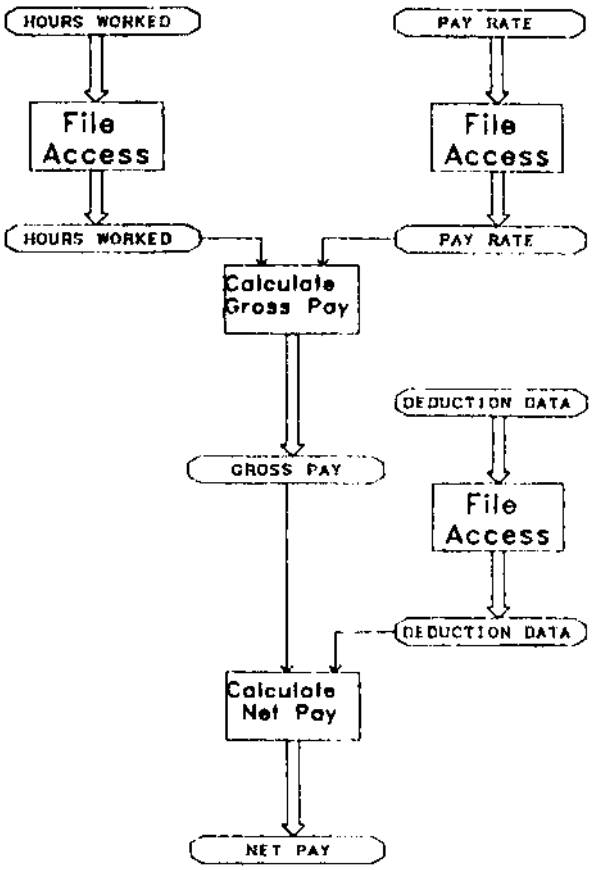

Figure 1 – A Sample Flow Diagram

base. Specialization hierarchies [6] are employed and a rich set of inter-object references exists. The system has been programmed in APL and uses a knowledge editor to build the objects (called ENTITIES) and their attribute descriptions (called SLOTS).

The knowledge base includes the four types of entities described below.

DATA - The domain specific data processed by the applications of interest.

DEVICE - Input, output, and storage devices which may be used in the application. Devices are related to the data they can access or produce. (File data, for example, can only be referenced by a particular class of devices.)

FUNCTION - Existing (pre-programmed) processing routines which operate on data relevent to the problem domain. FUNCTIONS reference the DATA they process and produce.

TRANSFORMATION - Summary knowledge of the possible data-to-data transformations. TRANFORMATIONs reference many other elements in the knowledge base including DATA (what is transformed to what), DEVICE (for input and output transformations), and FUNCTION (the process that does the transformation). As an example, a DATA-ACCESS TRANSFORMATION could be defined to retrieve stored information. The particular function employed might depend on the file structure of the data to be accessed. Multiple file structures could then lead to a specialization hierarchy of DATA-ACCESS TRANSFORMATIONS.

The flow diagram resulting from a particular consultation is represented in the same formalism and data structures as the static knowledge initially available to the system. (The initial knowledge structure is called the static knowledge base while the results of a particular consultation are called the dynamic knowledge base. As elements of the diagram are created they are associated with the most specialized frame description appropriate to that element. For example, a DATA-ACCESS TRANSFORMATION element may be created without yet knowing which actual access transformation will be employed. The constraint expression causing the creation is then re-evaluated in the context of any dynamically created DATA-ACCESS TRANSFORMATION elements. The resolution of constrained entity references (which is dependent on both the entity context for the resolution and on the dynamic state of the knowledge structure) has presented several challenges to the constraint resolution architecture.

IV THE CONSTRAINT ARCHITECTURE

Recent work has highlighted the role of constraints in processing frame or semantic network knowledge. (See, for example, [7].) Constraint processing has generally been used to limit search space, principally by applying constraints to a static collection of objects with the hope of finding only a subset which meet the constraints. The constraints to be applied come from several sources, the most common of which are the knowledge base itself and values for specific parameters generated within the consultation session.

Constraint processing is employed in our system to control the traversal of the knowledge structures and the selection of entities for inclusion in the flow diagram.
Constraints are used in building specialization hierarchies, and in specifying references from one entity to another. It is commonly found that these inter-connections between entities within specialization hierarchies become more constrained as one traverses the specialization paths; this permits specialization traversal over one type of object to implicitly constrain selection of a related object. (Knowing the data to be obtained, for example, will often constrain the choice of access mechanism.) An element admitted to the flow diagram may be associated with any level in a specialization hierarchy; subsequent discovery of

additional constraints (either from user interaction or specialization of a related element) then permits association of this element with increasingly more specialized entities in the knowledge base. In general, a consultation is complete when all elements of the flow diagram are associated with leaves in the specialization hierarchies, and all references made by these elements have been admitted to the diagram.

The application design task required an extension to the concept of constraint processing as a static selection process. Constrained entity references in the static knowledge base must produce instantiation of the appropreiate objects in the dynamic knowledge base, and cause the proper inter-entity references to be created. While it is possible for a single static reference to produce several dynamic instantiations (several instances a particular processing function, for example) it may also be the case that a suitable dynamic object already exixts and must be found when the constraining reference is resolved. Since correct inter-connection is fundamental to the flow diagram problem, a mechanism was needed not only to be able to state the general interconnection constraints in the static knowledge base, but also to instantiate the specific interconnections of the dynamically created flow diagram. This mechanism is described below.

The general form of a constrained entity reference is:

IS
entity-name
WITH
slot-name
(constraint expression)

where a constraint expression is a limitation on the value a slot can have. Examples of constraint expressions include:

```
= 6
ONE_OF ['red' 'white' 'blue']
IS entity_x
ONE_OF [entity_1 entity_2]
```

Whenever an entity is instantiated in the dynamic knowledge base slots with constrainted entity references (which are always dependent on the slots of the referenced entity) are evaluated in the dynamic entity context. As an example, consider an interconection in the flow diagram which is expressed as a mutual reference between the two connected elements. If slot1 of entity A was constrained to:

IS
entity-B
WITH
slot2
IS THIS-ENTITY-NAME

and slot2 of entity B was constrained to

IS
entity-A
WITH
slot1
IS THIS-ENTITY-NAME

then the processing of these constraints in the static knowledge base should provide a general limitation of acceptable candidates while processing these constraints in the dynamic environment must force the specific interconnection references. To accomplish this, two architectural principles were established.

1. References from one entity to another must cause the creation of a new dynamic object if the reference is from an existing dynamic object and the resolution is to a static object.

2. When processing a constrained reference to a dynamic object, the constraints are imposed on the target (constrained) slot.

Principle (1) causes the creation of the elements of the flow diagram, once an initial dynamic entity has been created. Consider the case where an instance of entity A has been created and slot1 of entity A is being evaluated. Its resolution will be a reference to entity B in the static knowledge base which will cause the creation of a dynamic instance of entity B, by principle (1). The constraint expression in slot1 of A (which further constrains slot2 of B to contain a pointer back to A) is now imposed on slot2 of entity B by principle (2). This effects the correct interconnection by mutal reference required in the flow diagram.

Several complications occur. To begin with, consider the case where the entity selection dictated by a constraint expression resolves to a list of potential candidates. Principle (2) requires that constraints be imposed on the dynamic object, but in this case no single object exists. We build a representation for this ambiguous object in order to record the constraint. These objects are used to represent the general class of entities that can satisfy the known constraints; the posting of new constraints may have the effect of redefining the class (to a subset) or of selecting a single entity from the class. This concept is critical to a consultation in which focus of attention can change before a fully instantiated form of the entity is determined. Our constraint architecture has been designed to permit maximum flexibility in representing classes of entities as the result of constrained searches.

Another problem occurs in the mutual reference case described above. We find here that the processing of slot1 in entity A requires the evaluation of slot2 of entity B which, in turn, requires the evaluation of slot1 in entity A. A looping situation is established and needs to be detected and properly processed. This requirement was one of several that led us to extend our concepts of the control flow mechanism in the consultation system.

V THE CONTROL ARCHITECTURE

The sequence of processing steps and its implication on focus of attention has been recognized as a critical issue in expert system design 8 . In most systems, control flow logic is cast in programming, a formalism not famed for its transparency or ease of modification. In systems with programmed control flow, developing a natural and and correct consultation requires a careful knowledge design and familiarity with the internal control logic. The recognition that the overall strategy for knowledge processing may reflect domain-specific expertise (e.g., when to look at what knowledge and what sequence of steps to follow) has led some architects to provide a formal knowledge representation for control [7,8]. This seems inherently correct and the system presented here reflects the authors' belief that control flow must be designed in a manner that permits modification by problem-domain experts, and that a suitable formalism must be found.

In our system the major processing is accomplished by a collection of PRIMITIVE ROUTINES designed to operate on our frame-structured knowledge representation. Examples of primitive routines are:

Create a dynamic object
Resolve slot semantics
Interact with users
Specialize a more general dynamic object

These routines do not communicate directly with each other; if one primitive determines that another primitive should be executed, it interacts with a SCHEDULING FUNCTION to schedule the execution of that task. Primitive routines to be executed are represented in an AGENDA which lists the identification of the primitive, the processing context in which it is to be run, and control information for the scheduling function. In our current implementation the agenda grows continuously with each new item scheduled and serves as our trace history information. Primitives may be scheduled for either dependent or independent execution. Dependent scheduling forces the complete execution of the scheduled primitive before the primitive requesting that scheduling is resumed. In the case of independent scheduling the order of execution is arbitrary.

Both the ordering of execution of pending agenda items and the automatic creation of agenda items (depending on the dialog state) reflect expertise that should be modifiable by and transparent to the problem expert. To effect this, antecedent-driven PRODUCTION RULES have been chosen to represent control knowledge. This choice was dictated mainly by our desire to permit non-programmers to enter and maintain expertise concerning the course (control flow) of a dialog. In particular, the current focus of attention for the dialog has often turned out to be properly a function of domain specific expertise rather than a result of domain independent, general dialog processing. This observation led us away from approaches such as attached procedures and toward more formal knowledge representations. The control rules may be invoked at either the time a primitive is scheduled or when a primitive terminates, and may reference the current agenda information as well as the general dynamic state of the consultation.

As an example, consider architectural principle (1), stated earlier, concerning the creation of a dynamic object based on the results of slot resolution. This piece of control logic is dictated by the particular task domain (flow diagram creation) and is reflected in an appropriate rule which is processed at the termination of the slot resolution primitive. In a similiar manner, the control logic to properly evaluate the constrained mutual references (described in the previous section) and inhibit looping is reflected in rules processed within the scheduling function. Specifically, when the agenda state reflects the occurence of a referencing loop the system forces the commitment to the mutual reference and processing continues.

Both the debugging and the modification of control flow in our consultations has beer, greatly facilitated by this design. Our experience, however, suggests several important future efforts.

VI THE SYSTEM STATUS AND FUTURE WORK

In the current implementation the control rules are implemented as APL code; this is to be changed to a formal data representation for the rules, coupled with a rule processing inference engine. The current knowledge base has been developed for medical sensor-based computing applications (the payroll example in this paper was chosen for its widespread familiarity), but has not been subjected to use as a production tool in a suitable laboratory environment. Sample dialogs have been run on limited test cases.

This system has been developed to ensure applicability to multiple task domains. To this end, generality has been introduced (e.g., in the control architecture) that might not have been required otherwise. It is our intent to apply the system to another task domain. Others we have considered include system configuration, and diagnosis (medical or otherwise). In this last case, we envision a major replacement of control rules to accomplish inferential diagnostic processing (as in MYCIN [9.1) , through the formalism of frame instantiation. Another interesting experiment would be to develop a MOLGEN-like system 7 , in which the control flow reflects a strategy of 'least commitment', instead of our current system which adopts a strategy of 'greatest commitment'. Once again, we envision the changes to be limited principally to the control rules.

VII CONCLUSIONS

The issues of control flow are critical to an expert system. Both task-specific and domain-specific expertise is involved, and a suitable formalism must be established for capturing and effecting this expertise. Antecedent production rules seem suitable to this task.

Frame-based systems are powerful problem solvers, especially when a general approach to constraint processing is taken. A system must have the ability to represent ambiguous results of constrained searches and to post further constraints to those representations as a means of possible disambiguation.

It is our belief that systems with mixed knowledge representations are easier to build and may be applicable to a wider variety of task domains. We further believe that a knowledge engineer should use representations that most naturally reflect knowledge to the appropriate expert, and that the use of multiple representations facilitates this goal.

Our current implementation has provided us with a sound experimental base for future development. The task of application design has forced us to address several important issues in expert system architecture, and is a potentially useful application of expert system technology.

REFERENCES

[1] Proceedings of a Conference on Application Development Systems, Data Base, Vol. 11, No. 3, 1980 (ACM order number 473800).

[2] Proceedings Application Development Symposium, GUIDE/SHARE/IBM, October, 1979.

[3] Reinstein, H.C. "Problem Solving in Frame-Structured Systems Using Interactive Dialog" In Proceedings AAA1, Stanford Ca., August, 1980, pp. 146-147.

[4] Minsky, M. "A Framework for Representing Knowledge" In P. Winston (Ed.) The Psychology of Computer Vision, New York, McGraw-Hill, 1975, pp. 211-277.

[5] Fikes, R. and Hendricks, G. "A Network Based Representation and its Natural Deduction System" In Proc. 1JCAI-77. Cambridge, Massachusetts, August, 1977, pp. 235-245.

[6] Bobrow, D. and Winograd, T. "An Overview of KRL, a Knowledge Representation Language." Cognitive Science, 1:1 (1977), 3-46.

[7] Stefik, M.J. Planning with Constraints, Ph.D. Thesis, Stanford University, 1980.

['8] Aikins, J.S. Prototypes and Production Rules: A Knowledge Representation for Computer Consultations, Ph.D. Thesis, Stanford University, 1980.

[9]| Shortliffe E.H. Computer Based Medical Consultations: MYCIN, New York, New York, American Elsevier, 1976.