2

MULTI SENSOR INTEGRATION

Deliverable No.: A003

Prepared by:
JAYCOR

Prepared for:
Naval Research Laboratory
Washington, DC 20375-5000

In Response to:
Contract #N00014-85-C-2044

12 May 1988

1. INTRODUCTION

Knowledge acquisition (KA) is the interactive transfer of domain knowledge from the expert to the program and the embodiment of this information in a computer understandable form inside the machine. Refinement is the process of verifying this domain knowledge for consistency and completeness. In addition to acquiring, KA sometimes involves discovering new knowledge, verifying it, and augmenting the existing knowledge base.

Knowledge acquisition is a major and important task in the development of an expert system. This is one of the more difficult tasks as there are no set methodologies to aid the process. Expert systems are knowledge based and their performance depends on the quality as well as the quantity of the domain knowledge. So far these expert systems are only in narrow domains with limited knowledge incorporated into them. This lack of generality makes them unattractive. For real applications, not only big chunks of knowledge are needed, a thorough understanding of the effects of adding or modifying a rule in a big set of rules is also needed. It is hard to comprehend such effects as the knowledge base becomes bigger and bigger. Knowledge engineering is a separate field in AI which involves acquiring knowledge, checking the knowledge for contradictions or errors, notifying and (suggestions for) correcting these errors, and finally augmenting the preexisting knowledge.

Presently knowledge acquisition is done through a knowledge engineer who interacts with the expert in the domain and the person who is developing the expert system. Acquisition of the large chunks of problem specific information required for real world application requires a lot of time and effort on the part of a high-priced knowledge engineer. Automating such processes saves time and effort. Finally research in KA in the long run might produce methods to automatically acquire knowledge which are superior to programming.

This report covers the various aspects of the Knowledge Acquisition problem. The covered topics are quite applicable to the Multi-Sensor domain in the sense that Knowledge Acquisition is a generic subject usable by all projects that require expert knowledge. The Multi-Sensor domain must meld information from a variety of information inputs, but the actual methods of that melding differ little from the single-source domain.

2. The Knowledge Acquisition Problem

Knowledge acquisition involves acquiring (or changing) domain knowledge consisting of facts, concepts, heuristics etc. Because it is a fairly complex process, a number of potential difficulties present themselves. These are itemized here:

(1) It is a costly and time consuming process. Experts are not cheap.

(2) Even if the expert is willing to co-operate most of the time it is very difficult to formalize how he/she makes decisions. Most of these experts don't have the inner knowledge of computers, representation of knowledge etc. They divulge their experterise in terms of their own knowledge representation which may be

hard to translate to a computer representation.

(3) The expert might forget to give all the details of the domain. The expert may tend to assume the existence of certain facts, for example, that objects released in a gravitational field tend to fall, a concept that must be explicitly stated but is ''obvious'' to the expert.

(4) As more and more knowledge is obtained it is highly probable that the heuristics the expert gives are not consistent or contradicting. Finding such contradictory information is a tedious job. Unfortunately, this usually happens while actually running the program.

(5) As knowledge changes with time, it [illegible] important to be able to make changes (add/modify) to the knowledge base and maintain the integrity of the system by keeping the changes in the KB consistent with the old knowledge.

3. Existing Tools

There are a number of existing knowledge-based tools which use different techniques to acquire knowledge. Some of these techniques are applicable to the Multi-Sensor Fusion domain.

EMYCIN

EMYCIN [VanMelle] is a tool to build knowledge based consultant programs. Actually EMYCIN is the domain independent version of MYCIN. Domain knowledge is represented as production rules and associative triples. The control structure is a goal-driven backward chaining of rules. Each rule and triple is associated with certainity factors to accomodate uncertainity of data. The syntax of the rules allow functions and predicates. It has a trace, explanation and verification mechanism imbedded in it. It is applicable for consultation and diagnostic systems with unreliable data.

EXPERT

EXPERT is developed in FORTRAN and is very efficient. This has evolved from Casnet/Glucoma. This is suitable for constructing consultation systems. Knowledge is represented as hypothesis, findings(data), and rules(inference). Hypothesis are inferred by the system. A measure of uncertainty is associated with hypothesis as well as with findings and rules. Control structure is data driven. It has excellent other facilities like explanation and trace mechanisms. It also has empirical testing capability in which the obtained results can be compared to the results of the stored data base of cases.

KASKAS evolved from PROSPECTOR. This is suitable for diagnostic tasks. It uses semantic networks to represent data. Objects form the nodes of the network and the links between them represent the relation between the objects. Inferencing is basically backward chaining. The method selects a top level node (hypothesis) and and recursively finds out the values of other nodes that effect the present node. It uses Bayesian techniques to propogate the probabilities. In some versions, best-first control structure is implemented which is slightly better than the backward chaining. In best first search, the system selects the next best node that can contribute to the solution of the problem and proceeds. Best node is decided by the expected yield and cost of determining the value of a node. It has a structure oriented editor which helps the expert add to the knowledge base. The system also provides default values to nodes.

ROGET

ROGET carries on a dialog with the user, finds the problem and gives advise as to which organization or architecture is suitable for the domain. Roget has some predefined conceptual structures to use once the type application is decided. (this is in theory. At present it has the conceptual structure of EMYCIN-like systems, i.e. diagnostic only). Once the application is decided, the representation, and inferencing frame work are taken from the stored internal conceptual frame work. With these predefined concepts, it guides the user with expectations to construct the knowledge base. Roget's architecture is also an extension of EMYCIN with task-block functions. Task-block functions build instances of clusters which represent facts.

KMSKMS (Knowledge Management System) is an interactive knowledge acquisition tool which help construct knowledge based decision systems. KMS provides a general conceptual frame work (attribute hierarchy) to represent domain knowledge. Features of the problem form the nodes of the attribute hierarchy, the links between the nodes represent the relationships between the features. Unlike the above mentioned tools, it supports three types of inference methods, namely Bayesian, Production Rule, and Hypothesize and Test. It has a text editor which helps the expert in writing the knowledge base.

TEIRESIAS

Teiresias constructed basically for EMYCIN is a knowledge acquisition tool. It uses meta-knowledge to check for errors, explain, and debug the knowledge base. Teiresias contains an object level representatin describing the external world (domain) and meta-level representation describing (actually monitoring) the object level knowledge. Davis's method uses model based understanding, i.e., for each concept that the system has to acquire instances of, a template is given containing what is needed to ask, how can it be acquired, what can be inferred, the type of the value to expect etc. These templates help Teiresias focus its attention on the present concept or context while acquiring the KB. This template based acquisition also allows the system to perform syntax and type checking. Meta rules allow the system to monitor domain rules, check their performance and change them if neccessary for efficiency, correct them if they are error ridden.

4. Refinement

Knowledge Acquisition also involves acquiring incremental knowledge and refining the existing knowledge. Most expert systems start with incomplete knowledge and improve this existing incomplete knowledge by adding new information and refining it. Two examples of correction, consistency checking, and refinement are the methods that the KES and EMYCIN systems use.

In KES once the expert creates the knowledge base using the editor, he runs the program. The program then does some type checking and checks for syntax errors, and proceeds only if the resulting KB is error free. If errors are encountered, the program gives a message to that extent and stops. The user can correct them and run the program again. Even while running the program, KES checks for the validity of user input to some extent. Some consistency checking is provided. For example, the sum of conditional probabilities must be equal to one and if it is not it complains. Other than these minor things, it can't do any semantic checking.

In EMYCIN, some semantic checking is provided along with the syntax and type checking. For example, while acquiring the knowledge base if an undefined name is popped out somewhere it reports that to the user and lets the user define that variable. EMYCIN has an elaborate front end which helps the user to enter knowledge. Its trace and explanation facility allow the user to browse through the execution and find other logic errors.

5. Issues In Designing A New Generalized KA

There are three kinds of expert systems tools[Lee]. The first one is the "low level" tool in which a knowledge representation and inference are supplied as a tool, e.g. OPS5, RlI. In this kind the user has to represent the domain knowledge and give the control structure explicitly. The second one is a "general purpose" tool that provides higher capabilities but the user still has to map the domain problem onto these capabilities. The third one is a "highly structured" tool like in EMYCIN and is a very powerful tool to construct the knowledge base systems. The problem with a highly structured tool is that it works only for a fixed type of problems. Thus the generality is lost.

All of the tools described in the preceeding sections work for a particular application and would come under the third category of highly structured tools. What we are interested here is not in a generalized expert system construction tool, rather a generalized knowledge base acquisition tool to acquire an error free knowledge base which is consistent and complete, so it can be used in different applications. KA tool should represent the acquired knowledge in some internal form suitable for the application part to use it. So the type of representation is dependent on the application of the type of the domain problem. As such, there is no separate tool like a KA tool which can stand by itself. We present this argument as we look at some of the features, problems and solutions of a generalized KA tool.

The general KA tool should

1 acquire the domain knowledge from the expert and represent it in some suitable internal form (acquisition part).

2 find and correct errors (correction part).

3 insulate the expert as much as possible from the organization and internal details of the computer.

5.1. To Acquire and Represent Knowledge

A number of different items are required for getting and representing the knowledge. These are enumerated below.

1 interaction: communication between the user and the system (front end)

2 organization: organization of the acquired knowledge for correct and coherent behaviour

3 explanation: should provide explanations.

4 accommodation: should accommodate new knowledge

5 Reformulation: reformulating existing knowledge to apply in new circumstances

6 evaluation: should be able to evaluate its performance

7 compilation: should be efficient.

5.2. Interaction

Acquiring knowledge from an expert involves some kind of communication between the program and the user. So a good user friendly interface is needed. There are 4 types of them:

1 menu driven

2 query driven

3 graphical

4 natural language

Sometimes knowledge based editors are also used to interact with the user. Interfaces, besides acting as a communication agent between the program and the user, should also do some initial error (mostly syntactic) checking. Natural language interfaces are stylistic but are the most complex of all.

5.3. Organization

Organization involves selection of appropriate representation suitable for the domain. The KA tool is envisioined as a tool with which the expert can transfer his knowledge into an internal (representaiton) form. The expert here is assumed to be a naive computer user. So the program should guide him, in transfering the knoweldge. In order to do that the program should know what to ask, and what to expect.

In other words, knowing the representation the program can capitalize on the strongly typed nature of the underlying data structures. Besides these, the program has to correct the knowledge base. All these things are dependent on the organization of the conceptual structure of the domain application as different conceptual structures are suitable for different problem types. EX. for diagnostic problems - backward chaining with production rules or hierarchical networks. understanding - blackboard architecture with frame representation.

Error checking is of two types: syntactic and semantic. They are discussed elsewhere in this section. What we are going to see is how it is going to dependent on the reprsentation of the KB. The semantic error checking involves finding semantic errors not only with the given factual knowledge but also with the other inferential facts implied from the given knowledge. So the program should understand the semantics of the knowledge base as to what a fact means and what would that imply. In other words the rules of the game should be placed on the table before playing and deciding if a move is correct or wrong.

This leads to the problem of deciding which representation is appropriate. General representations are frames, semantic networks, first order predicate calculus, and production rules. All these methodologies are based on different assumptions about the nature of the domain and all of them have their premise. Each of these representations go with different kinds of inference engines which combindly are suiatable for different applications. So choosing a representatin also depends on the type of inference engine suitable for the domain application. So selection of suitable representation is complex and there are two ways to deal with the selection problem.

In the first method different representations or conceptual structures are available to the program. The program after deciding the type of the domain application chooses one of the internally defined (stored) conceptual frame work suitable for the domain and proceeds with the acquisition of the knowledge. This frame work guides the program as to how to represent the acquired knoweldge and what to expect during the dialog with the user. Ex. Roget has different internal conceptual frame works and chooses one of these depnding on the type of the problem and proceeds. KMS has 3 types namely: Bayesian, production rule, hypothesize and test. The user can choose one of these that is best suitable for his application and then the program acquires and represents the knowledge base and performs some minor structural dependent error checking like the sum of the conditional probabilities should not be greater than 1 in the Bayesian method.

The difficulty with this type of approach is that the program has to keep a number of conceptual frame works inside. Each frame work has its own inferential methods and error checking. The program becomes big as new conceptual stuctures are added. Roget's aim was in this direction, but it has the capability of constructing the KB only for diagnostic domains. For other applications, it simply gives advice as to which organization is suitable for the domain problem and leaves it there. There is no error checking incorporated in this. Rogets has a nice feature of giving examples to the user in helping the user decide which type of application the domain porblem is.

The second method suggests that a common representation that is suitable with different types of inference methods shoudl be devised and used. ie. select a common representation that has the features of all the above conventional representation and represent the acquired domain knowledge in that irrespective of the domain application.

Several general methods are suggested to incorporate all the features of the above mentioned representations. Levesque suggested having a representation characterized by the functionality and not by the structures they use to represent the knowledge. Weiner and Palmer suggested abstract data types which include features common to the above mentioned general representation methods to represent domain knowledge. Another method suggested by Charniak is to combine frames and logic to represent a variety of problem domains. Waterman identified that combining certain pairs are easy and expressed concern about the problems of intergrating multiple forms of control that might arise by integrating more representations.

Basically domain knowledge consists of two components. The structural component consisting of facts, observations etc. which represent the fixed assumptions of the expert about the nature of the solution. The second component (inferential) describes how certain facts follow from particular observatoins and indicates when these inference should be performed.

The most suitable form to represent the factual knoweldge is frames as it is highly structured, modular and allows inheretence. The inferential compound is represented by a set of rules in most cases. These rules use the structural component of the domain problem representation to refer to objects in their premise and action parts. Rules are well suitable to represent the inferential part as they are modular, and their ability to represent small parts of reasoning and accomodate changes. The type of solution method (inference) depends on the domain, and so the system should have a variety of inference mechanisms like forward chaining, backward chainig, island driven, goal driven, data driven, even driven etc. For general organization, frames combined with logic and production systems is highly suitable. This allows a common representation of the data which can be used with any selected inference methods. The type of inference method depends on the type of application. One interesting research topic to that extent is to combine logical programming like prolog with frame representation. Ability to incorporate and deal with meta knowledge is certainly a desirable feature. Meta knowledge allows some kind of domain dependent checking of consistency and compleness of the knowledge base, besides the incorporation of some control knoweldge.

Now that the representation problem is taken care of lets get back to our original problem of "*what to ask, what to expect, how to correct errors*". All three of them are addressed by Davis. The first two are dealt by using template based acquisition. In this approach each concept or object has a template attached to it(meta-information) , which carries all the information about the object as to what should be asked, what are the legal values etc.. The template is a domain dependent structure, which can be acquired from the user. Some type of error checking (syntactic) is done while acquiring the knowledge using these temples. Domain dependent semantic errors can be checked and corrected using by meta rules given by the user, which specify the completeness and consistency of the domain KB. EX. if there is an operation to be performed on a machine and if that machine is not in the kb then the kb is not complete and ask for the details of that machine.

5.4. Explanation

A module giving explanation as to why it is asking question, or why it thinks a certain information in the KB is contradictory etc is strongly desirable as it allows the expert to understand and recognise errors.

5.5. Accommodation

Once the knowledge base is acquired, the system should be able to acquire and augment new knowledge. Actually discovering new knowledge from experience and augmenting it to the KB is desirable. This is done in Davis Teiraseas using meta-rules. Some times the domain expert observes new heuristics and facts which should be augmented to the existing KB. This should be made possible. Some times as new knoledge is added it is desirable to change the structure of the KB.(say the inference method). The system should be able to do this without changing the representation. This can be achieved by using a uniform representation of knowledge irrespective of the inference methods used.

5.6. Evaluation

The system should be able to analyze its performance and guide the expert in locating unnessesary rules or rules that are being fired frequently, situations reaching a deadend that could never proceed furthur due to lack of data, etc.

5.7. Compilation

The system should be able to locate freqently fired rules and compile them as to improve the system performance.

5.8. Correction Part

This is a major part in Knowledge Acquisition. To keep the integrity of the KB, the system first has to find errors, and then should either report thrm to the expert or rectify them if possible. Errors are mainly of two types: syntactic and semantic

Syntactic errors are those that arise by faulty spelling or illegal syntax. These can be corrected by using knowledge based editors. They include syntax errors and finding undefined variables. Some allow the continuation of the acquisition process and allow the user to define them latter. If the user forget to define the variable the system usually warns the user of these variables. Other systems require the user to define the variable immediately and then proceed with the acquisition.

Typechecking is done with the values of all variables. It checks the values with the legal range of values the attributes can have and warns the user if they are not so. Some systems won't even take other values.

Semantic errors are logical errors which arise due to an inconsistent or incomplete KB. The KA tools, in addition to syntax and type checking, should also see if the KB is consistent and complete. Generally, semantic checking is done by observing errors in test cases. A number of test cases should be run in order to find semantic errors.

This is tedious, time taking, and tends to be unreliable. Some systems use a method where they can automatically generate examples, solve the problem, analyse the results for errors and rectify them. This is a cyclic process and the system settles down after several runs. The problem with this is that the system has to run the problem to find mistakes. What we are interested here is finding problems in a given KB without actually running system for results. Automatic checking of the KB is possible for the program if the user can give meta rules as to what is an error, contradiction or incompleteness in the domain.

Checking for consistency of the Knowledge Base can detect a number of kinds of errors in general: Conflict -- succeeding two rules in the same situation with conflicting results, Redundancy -- succeeding two rules in the same situation producing the same effect, Subsumption -- succeeding two rules with different conditional parts, one of which is a part of the condition part for the second rule; This restricts the other rule and thus produces redundancy, the well known problem of Infinite Loops -- getting into an infinite loop so the program won't proceed any further; These can be detected by knowing the syntax of the rules and by observing the rules.

Completeness also is very important but is difficult to check for. It is in essence the missing ofrules in certain situations. Even though it is logical, it is impractical to have rules for all possible situations. Nonetheless missing rules can be spotted by checking for rules for all possible situations that arise by enumerating all attributes.

Consistency and completeness to a certain extent are independent of the KB, but to find out all the errors one need to know about the domain. As explained earlier meta rules provide a good way of testing KB for errors.

There are many other Knowledge Base refining methods. The LEX problem solver [Hayes-Roth] is another good program which checks its KB and control knowledge for logical errors. Lex has 4 modules. The first one solves the problem. The second one critiques about the solution and analyses it. The third module produces new generalized rules according to the critique done in the second module. The forth module generates new problems providing training data to test these refined rules. It stores only maximally specific and maximally general versions of rules instead of storing all refined rules at each cycle.

Hayes-Roth's TL (Theoretical Learner) method rectifies error ridden knowledge and extends the range of its applicability by generating new concepts. The method first localizes the problem area, then debugs the solution plan by fixing the knowledge that is responsible for the fault.

Winston's augmented rules are another method. In this method rules have an additional UNLESS clause besides the if & then. This method suggests the inclusion of a predicate that can represent the failed situation in the UNLESS clause of the failed rule. This sometimes creates problems like the barring of these rules by other existing rules. Therefor Winston devised a method in which other rules obstructing the present rule are found and barred from firing. This is also called a sensor.

6. Concluding Remarks

A number of different issues of the Knowledge Acquisition problem have been discussed in this report. The reader is referred to the references which provide a much deeper review that this survey can provide. The Knowledge Acquisition issue is an ongoing research topic, one present in all domains in addition to the Multi-Sensor one.

References

[1] Reggia A. James., "*KMS Manual*", University of Maryland, Jan, 1982.

[2] Motoi Suwa, Carlisle A. Scott, Shortliffe H. Edward., "*An Approach to Verifying Completeness and Consistency in Rule-Based Expert System*", AI Magazine, Fall 1982.

[3] VanMelle J. William, "*System Aids in Constructing Consultation Programs*", Ph.D. Thesis, CS Department, Stanford University, 1980.

[4] Levesque H.J., "*Foundations of a Functional Approach to Knowledge Representation*", AI Journal, Vol 23, Number 2, July 1984.

[5] Charniak E., "*A Common Representation for Problem-Solving Language Comprehension Information*", AI Journal, Vol 16, Number 3, July 1981. Politakis Peter adn Weiss M. Sholom., "*Using Empirical Analysis to Refine Expert System Knowledge Bases*", AI Journal, Vol 22.

[7] Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell., "*Machine Learning, An Artificial Intelligence Approach*", Tioga Publishing Company, 1983. Jack Mostow., "*Workshop Review, Principles of Knowledge-Based Systems*", SIGART Newsletter, ACM, April 1985.

[9] M.Suwa et al., "*Knowledge Base Mechanisms*", Fifth Generatin Computer Systems, pp 139-146, North Holland Publishing Company, 1982.

[10] M. Amamiya et al. "*New Architecture for Knowledge Base Mechanisms*", Fifth Generatin Computer Systems, pp 139-146, North Holland Publishing Company, 1982.

[11] Lee Brownston, Robert Farrell, Elaine Kant, Nancy Martin., "*Programming Expert Systems in OPS5*", Addison-Wesley Publishing companay, 1985.

END
DATED
FILM
8-88
DTIC