IFAC/IFORS/IIASA/TIMS

The International Federation of Automatic Control
The International Federation of Operational Research Societies
The International Institute for Applied Systems Analysis
The Institute of Management Sciences

SUPPORT SYSTEMS FOR DECISION AND NEGOTIATION PROCESSES

Preprints of the IFAC/IFORS/IIASA/TIMS Workshop
Warsaw, Poland
June 24-26, 1992

Editors:

Roman Kulikowski
Zbigniew Nahorski
Jan W.Owsiński
Andrzej Straszak

Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland

VOLUME 1:

Names of first authors: **A-K**

SYSTEMS RESEARCH INSTITUTE, POLISH ACADEMY OF SCIENCES

SUPPORT SYSTEMS FOR DECISION AND NEGOTIATION PROCESSES
Preprints, IFAC/IFORS/IIASA/TIMS Workshop, June 24-26, 1992, Warsaw, Poland

KNOWLEDGE ACQUISITION SUPPORTED BY KNOWLEDGE BASED SYSTEM

Cezary Iwański

Systems Research Institute, Polish Academy of Sciences
ul. Newelska 6, 01-447 Warsaw, Poland

Abstract. In the paper an Interactive Knowledge Acquisition System IKAS is presented. This system consists of two parts: procedural and declarative. An ordinary expert system shell Sokrates is a procedural part of the system. A Meta Knowledge Base (MKB) including skills how to elicit knowledge composes a declarative part of IKAS. The acquired knowledge can be immediately verified, tested and used in expert system based on Sokrates shell.

Keywords: knowledge acquisition, expert systems, knowledge based systems.

1. INTRODUCTION

Building an expert system involves eliciting the knowledge that human experts use when solving problems. Experience has shown that the process of knowledge acquisition is a crucial issue in the construction of virtually all knowledge-based systems. This process is time consuming, difficult and often a major bottleneck in the production of expert systems.

At the beginning knowledge engineers used classical methods to elicit knowledge from experts, eg., interview or "brain storming" as well as more sophisticated methods applied earlier in clinical psychology, eg., "card sorting" or "multidimensional scaling" etc. (cf. Politakis et al. 1984).

The ways to automation of tbis process have been looked for. Machine learning algorithms bave been sometimes used to help. Among various machine learning approaches, inductive learning (cf. Dietterich et al. 1981, Iwanski & Szkatula 1991, McDonald et al. 1989, Michalski 1983) as well as genetic algorithms (cf. Booker et al. 1989) have been applied most often. However the constructed rules could be used only to classification-like problems and the learning process demands a lot of examples in order to learn the rules.

For the last few years some automatic knowledge acquisition systems have been developed such as: MORE, MOLE, SALT, KNACK, SIZZLE and RIME. They ask questions to the user and carry on a sophisticated process of knowledge elicitation. The excellent survey of those methods has been done by Sandra Marcus (1988). They are very useful procedural tools supporting strongly knowledge acquisition processes.

IKAS is an automatic knowledge based acquisition system. It consists of two parts: procedural and declarative. An ordinary rule based expert system shell Sokrates is a procedural part of the system. This system can be used independently as a standard

expert system if any knowledge base is involved in it. A Meta Knowledge Base (MKB) including knowledge how to elicit knowledge from experts composes a declarative part of IKAS. The whole acquisition process runs in accordance with the General Knowledge Acquisition Scheme (GKAS, see section 3) determined by MKB.

The approach where knowledge acquisition process is controlled by knowledge based system has a number of advantages. Any changes to improve this process can be quickly inserted to the system. Knowledge base can be easily extended by specialized portion of knowledge, eg., how to elicit knowledge from any domain of medicine, etc.

2. EXPERT SYSTEM SHELL SOKRATES

Sokrates is a user-friendly rule-based expert system shell. It comprises two main modules: knowledge base editor and inference engine with explanation facilities.

Editor enables: to insert new elements (facts or rules) to base, to delete elements from the base, to change any features of elements, to save data on disk, to read data from disk into working memory, to print whole base or single elements and to browse the base.

Inference engine comprises forward, backward and mixed chaining strategies.

The base consists of facts and rules. There are some distinguished facts called special facts. They can only be used in the consequent part of rules making the system perform an action. There are 10 special facts. Some of them are described below.

A fact can be defined as a triple (n,v,P), where n is a fact name, v a fact value and P a set of parameters. There are various parameters which include information needed during inference, e.g., type of fact (nominal, numeric, fuzzy), fact status (goal, subgoal, transfer, etc.). Some parameters are very useful for communicating with the user, e.g., ask contains a question to the user for value of the fact, instr contains instruction for the user, etc. If any information was needed to be displayed for the user the following rule could be fired:

IF
premises are fulfilled
THEN
...
instr fact_name
...

The special fact **instr** makes the text contained in parameter instr of the fact, named fact_name, be displayed. There are similar special facts for other text parameters, too. By means of these special facts communication with the user is flexible and depends on the previous dialogue.

Another important special fact is **control**. One of the values it can take on is read_base. The using of this fact results in locating a new subbase of knowledge at working memory. The transfer facts (status transfer) from base previously occupied working memory will be merged to the new read base. This process is called dynamic base reading. It allows to divide a knowledge base into subbases. The subbases can be gradually tested, verified and improved.

There are also other special facts defined in Sokrates as, eg., **call** which enables to call procedures.

From knowledge acquisition point of view another internal utility module is important. This module contains procedures operating on lists. It is not valid for the user but some

of these procedures will be used to explain GKAS described below.

3. KNOWLEDGE TO ELICIT KNOWLEDGE

The second part of IKAS system is the Meta Knowledge Base (MKB) including knowledge how to elicit knowledge. The whole acquisition process is controlled by this knowledge and runs accordingly to the scheme called General Knowledge Acquisition Scheme (GKAS):

1. **INITIALIZATION**;
2. **VERIFICATION**;
3. **USER_LEVEL**;
4. **INFORMATION**;
5. **GOALS_DEFINITION**;
6. `Main loop of the scheme;`

```
while list_of_goals ≠ nil begin
    goal:=HeadCut(list_of_goals);
    MAKE_RULES(goal);
    while list_of_rules ≠ nil do begin
        rule:=HeadCut(list_of_rules);
        list_of_ante:=Ante(rule);
        PutToBase(rule);
        while list_of_ante ≠ nil do begin
            ante:=HeadCut(list_of_ante);
            MAKE_RULES(ante);
        end;
    end;
end;
```

In the above scheme 4 different quantities can be found: lists (3 items), elements of lists (3), list procedures (3) and processes (6). The meaning of lists, elements of lists and procedures is quite clear. Processes are described in detail in the following subsections.

3.1. INITIALIZATION

The first subbase which must be read to start acquisition process is called *initialization*. It contains rules which ask the user preliminary questions as, e.g., "Is it your first contact with IKAS system ?", "Do you start to build knowledge base or you want to continue the interrupted process or you want to improve the already existing base ?".

If the user answers that he/she wants to improve the existing base the rule reading subbase *improving* into the working memory will be fired (see section 5).

If the user answers that he/she wants to continue the interrupted knowledge acquisition process then the system, after some extra questions, resumes the process at the point where it stopped (see section 4).

Now we proceed the situation when the user starts to build a new knowledge base. After his/her answers are given the rule reading subbase *verification* into working memory is fired.

3.2. VERIFICATION

The main goal of this step of GKAS is to establish whether the application of expert systems technology to user's problem can bring a success. A subbase contains many questions which can be found in literature on expert systems. A system wants to know, eg., if there exists classical/standard approaches to the problem, if the problem is appropriate for ES technology (narrow domain, specific knowledge, uncertain and not well formalized/systematized knowledge), if there are experts disposed to cooperation, and so on.

After the conversation the system gives the suggestion whether it is a good idea to build an expert system for the user's problem.

If the conclusion of the interview is positive or the user insists on building expert system then new parts of MKB called *user_level* will be transported to the working memory.

3.3. USER_LEVEL

In this step the user is not only asked but also taught if needed. The sequence of questions is given, eg., "Do you know what expert system is", "Do you know any world famous expert systems ,eg., DENDRAL, MYCIN, PROSPECTOR or R1", "Do you know any knowledge representations", and so on. If the user does not know the concept then the appropriate definition or information is displayed.

The goal of this step of GKAS is to establish the user's level of knowledge about ES technology. It influences the type of questions or explanations given in the sequel to him/her by the system.

3.4. INFORMATION

In this step the user can obtain information about system IKAS itself. If not needed this point can be omitted.

3.5. GOALS_DEFINITION

The next step is to establish possible final conclusions (hypotheses) the system built by the user will be able to reach. The questions are similar to this one "How could you call fact-hypothesis which system could reach as a final conclusion ?". After receiving the name of hypothesis the system asks for the value of this fact and unknown parameters. Some parameters are obtained automatically, eg., status goal. Finally, the fact is appended to the list_of_goals.

Then the user is asked whether he/she wants to introduce the next hypothesis. If "yes" the next hypothesis is defined if "no" the main loop of the GKAS begins.

3.6. MAKE_RULES PROCESS

The fundamental part of GKAS is the process called **MAKE_RULES**. It consists of two stages. First, the set of rules verifying possible occurrence of a given fact is created. Then the set of rules negating the occurrence of the fact is constructed. Below the first stage is described.

Let F* denote the fact which should be verified.
ante(R) - premises of rule R supported fact F*

```
end1:=false;
repeat
    ante(R):=∅;
    F:=SUPPORT(F*);
    ante(R):=F;
    end2:=false;
    repeat
        "Does the rule ante(R) → F* is correct and complete ?"
        If "yes" then begin
            FINISH_RULE(R);
            Push_on_Stack(R);
            end2:=true;
        end;
        if "no" then begin
            G:=NEXT_SUPPORT(F*);
            ante(R):=ante(R) ∩ G·
        end;
    until end2;
    "Are there other facts which can explain the occurrence of
    the fact F* ?"
    if "no" then end1:=true;
until end1;
```

There are 3 processes and one procedure in the scheme above.

SUPPORT(F*) is a process in which the system asks the user "What fact can explain the occurrence of the fact F* ?". The user defines fact F by giving its name, value and some unknown parameters. The defined fact is appended to the knowledge base being created (not to MKB).

The **FINISH_RULE** is a process during which the system asks for the name of the rule R as well as for its certainty factor.

The list procedure Push_On_Stack(R) is called to append the rule to the list_of_rules.

During the **NEXT_SUPPORT** process two questions are put forward. The user is asked "What next fact should be added to the rule premises ?" and "what kind of changes should be done in the premises ?". With respect to the user's answer the new fact can be added to the premises, some changes can be done (names, values) or one of the facts can be deleted from the premises.

The internal `repeat-until` loop is finished (end2=`true`) if the antecedent part of the rule is ready. The main `repeat-until` loop is finished (end1=`true`) if the set of rules confirming occurrence of the fact F is ready.

The second stage is very similar to the first stage. Roughly speaking the processes **SUPPORT**(F*) and **NEXT_SUPPORT**(F*) are replaced by **SUPPORT**(¬F*) and **NEXT_SUPPORT**(¬F*) processes.

4. INTERRUPTING AND CONTINUATION OF ACQUISITION PROCESS

Any inference session can be interrupted in Sokrates by pressing Alt+S key. Since the knowledge acquisition process is understood here as an ordinary inference session it can be interrupted at any time as well. After pressing Alt+S the system asks the user if he/she wants to start the inference process from the current stage in the future. If the answer is "yes" then the system saves the current state of the new knowledge base as well as the current state of the MKB.

5. KNOWLEDGE BASE IMPROVEMENT SCHEME

The Knowledge Base Improvement Scheme is involved in IKAS system. Because of the shortage of place KBIS can not be described in detail. It enables to change or delete any rule from the new knowledge base without distortion of its global structure.

6. MAINTENANCE OF KNOWLEDGE BASE

During the whole knowledge acquisition process the Knowledge Base Maintenance System (KBMS) is active. There are some procedures which check if the knowledge is consistent. They check whether the same fact does not infer fact A and its negation $\neg$A, and whether circles, of type F$\rightarrow$ ... $\rightarrow$F, do not exist.

The completeness actually follows from the general assumptions of GKAS and KBIS, because each fact (except leaf facts of course) has at least one explanation.

7. CONCLUDING REMARKS

IKAS is a knowledge based system which supports knowledge acquisition process. The acquired knowledge can be immediately tested and verified by means of Sokrates shell and eventually used as expert system.

LITERATURE

Booker L.B., Goldberg D.E., Holland J.H. (1989): Classifier Systems and Genetic Algorithms, Artificial Intelligence, Nr. 1-3, Vol. 40, ss. 2235-282.

Dietterich T.G., London R., Clarkson K., Dromey R. (1981): Learning and inductive inference. W: Handbook of Artificial Intelligence, Cohen D., Feigenbaum E. (red.), Kaufman, Los Altos, ss. 323-511.

Iwanski C., Szkatula G. (1991): Inductive Learning supported by Integer Programming, Computers and Artificial Intelligence, Vol. 10, No 1, pp. 57-66.

Marcus S. (1988): Automating Knowledge Acquisition for Expert Systems, Kluwer Academic Publishers, Boston.

McDonald B.A, Witten I.H. (1989): A Framework for Knowledge Acquisition through Techniques of Concept Learning, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No 3, 499-512.

Michalski R. (1983): A Theory and Methodology of Inductive Learning. W: Machine Learning, R.Michalski, J.Carbonell, T.M.Mitchell (red.), Tioga, Palo Alto.

Politakis P., Weiss S.M. (1984): Using emperical analysis to refine expert system knowledge bases, Artificial Intelligence, 22, ss. 23-48.