Sci. J. Fac. Sci. Menoufia Univ. Vol. VIII (1994)

ACE: ADVISOR CONSULTING EXPERT SYSTEM

By
Dr. H.M.M. Harb
Azhar university, Faculty of Engineering,
Computers and Systems Engineering Department

ABSTRACT

This paper presents an expert system which advises the graduate student to select the thesis advisor such that the student's point of research and the advisor research interests are most compatible. Different degrees of research point specifications are allowed, the more specification is considered, the closer the advisor(s) for the point are selected. ACE has two modes of interfacing: the query/response and the keywords interfaces.

The research areas are given as a major tree where each node represents a major (research area) and the user can navigate through the tree. Implementation considerations toward the knowledge base representation and the interface engine are included.

Keywords
Expert Systems, Query/Answer systems, Data Structure, Logic Programming.

1. INTRODUCTION

Artificial intelligence attracted many computer scientists in the last two decades where AI systems have already been introduced in many areas such as natural language processing, automatic programming, robotics, theorem proving and expert systems. An expert system is a consulting program that exhibits a degree of intelligence similar to that of a human expert in a narrowly defined field. Expert system implementation areas are such as financial budgeting, air traffic control, medical diagnosis, mineral explorations and computer configuration. This paper presents the design and implementation of ACE system that may be consulted by the graduate students to select their advisor(s). The three major components of the expert system are the knowledge, the user interface, and the inference engine. A user should be able to interact with the expert system easily.

The flexibility, richness and representation of the knowledge base constitute the main power of the expert system. The knowledge may be represented by rules, frames, semantic nets or predicate calculus. The most suitable representation model differs from one problem to another depending on the application field. For the rule-based expert system, the production system, the knowledge base are represented as rules and facts.

Inference engine requires a search through the knowledge base to use and/or to modify it. There are many searching policies. Depth first and breadth first are most common searching techniques for AI. The inference engine may have the [illegible] explaining its reasoning and learning to acquire new knowledge and modify the old knowledge.

Logic programming is usually used to implement expert systems. Prolog is a common logic programming language for this purpose since it is powerful to represent and manipulate trees and lists. Prolog has also the built-in backtracking facility. The above two features of Prolog, ease of tree and list representation and depth first with backtracking built-in searching technique, explain our selection of Prolog to implement ACE.

Section 2 discusses ACE model organization and its knowledge base. Design and implementation are considered in section 3. Section 4 concludes the paper and suggests some extensions for this research.

2. The ACE Model

The graduate students with thesis option usually come up with their points of research and most of them do not know the staff research interests. ACE plays an important rule to help the graduate student in suggesting thesis advisors with research interests close to the student's point of research.

The main job of ACE is to come up with a list of advisor names and their research areas which are the most compatible with the student's point of research. We may refer to the point of research of the student as his/her major.

2.1 The ACE Model Organization

The ACE organization may be divided into three main modules: the knowledge base module, the inference engine and the user interface module. The knowledge base (KB) describes the majors (the research areas), the attached lists of advisors and the attached list of keywords and they are called major knowledge base (MKB) andadvisor knowledge base (AKB), keyword knowledge base (KKB) respectively. The knowledge base representation is discussed in section 2.2.

The user interface (UI) describes how a graduate student (user) interacts with ACE. The UI consists of two submodules, each represents a different way of interaction. The two sub modules names come from the functions they do. The first sub module is query/response user interface (QUI) and the second is keyword user interface (KUI).

The inference engine (IE) function depends on the user interface mode. IE makes use of the built-in backtracking feature of Prolog programming language. In case of KUI, IE uses depth-first search (DFS) technique. For QUI mode IE matches the input user response with data in MKB. The result of matching process (MT) gives affirmative or negative results consistently. The ACE inference engine will be discussed later. IE has a help facility to print majors, keywords, and advisors available (PR). Fig. 1 shows the main modules of ACE and the submodules of each module.

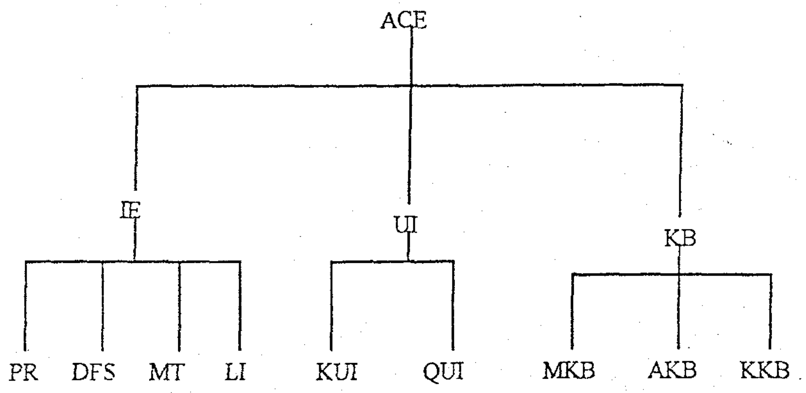


Figure 1: The ACE Model Organization

2.2 The ACE Knowledge Base

The knowledge base of this model is built for the majors and the corresponding research areas in the computer science department as a case study. In other words, ACE, as has been implemented in this paper, may be consulted only in this restricted knowledge base, but it may be extended to other research areas (other departments) by considering the appropriate knowledge base. The major knowledge base may be thought of as a major tree. A node in the tree represents a major (research area). Child nodes represent the sub-majors of a major represented by their parent node. Fig. 2 shows a sample of the major tree. The major abbreviations and the corresponding meanings are listed in Appendix A. There are no restrictions of the depth and width of the tree, so the major tree may be expanded to any degree of details. Inspecting the major tree arises comments as it follows. A (sub)major node may be expressed as a full path name starting from the root, for example cs/sw/is/sa is sa sub major. There may be a sub major attached with more than one major. The ddb, for example, is recognized as a sub-major of ds (cs/sw/ds/dds) and it is also a sub major of db (cs/sw/is/db/dds). There is no need to show the sub majors of ddb in the knowledge base twice (if they are completely identical) and we may also note that the advisor lists attached with ddb nodes (ds/ddb node and db/ddb node) should be identical. The above note also applies to the attached keyword lists as it will be discussed.

For each major (node in the major tree), there is an attached list of advisors interested in it. The attached advisor list with a child node is a sublist of the list attached with its parent. In other words the list attached with a node is a union of all sublists attached with its child nodes. For example the list of advisors interested in distributed database (ddb) is a sublist of the list of advisors interested in database which in turn is a sublist of the advisors interested in information systems(is). It is assumed that there may be overlapping between the advisor lists attached with different nodes. This assumption is based on the accepted fact that an advisor may be interested in more than one research area. For example the intersection between the advisor sublists of sa and ddb may not be empty. It is also assumed that the intersection of attached keyword lists with different nodes may not be empty.

3. The ACE Design and Implementation

As has been discussed in section 2, the knowledge base of ACE is presented as a tree representing the research areas (majors) and two lists associated with each node (the advisor list and the keyword list). So the system is implemented in Prolog for its capability to represent trees and lists and for its built-in backtracking searching technique that is helpful for tree searching. Fig. 3 shows the ACE basic modules with three module types: programming modules the disk resident knowledge base modules, and the memory resident (temporary) knowledge base modules. The programming modules are keyword user interface, depth first searching (DFS), user interface (UI), help messages printing (PR),

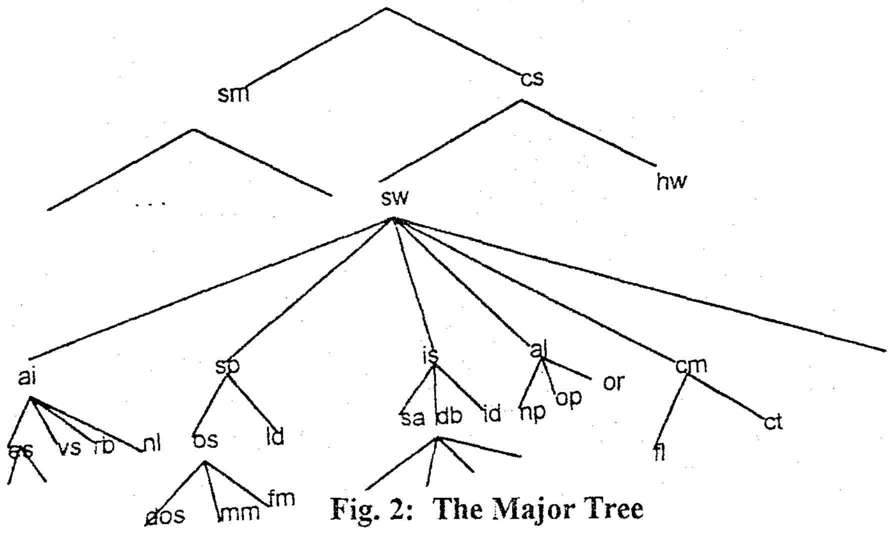


Fig. 2: The Major Tree

the associated advisors list (LI), user windows set up (SW), query user interface (QUI), and match of the user submitted major with the data in KB (MT). The disk resident knowledge base modules are major knowledge base (MKB), advisor knowledge base (AKB) and keyword knowledge base (KKB). The memory resident knowledge are the working knowledge (WK) and they are temporarily kept in the primary memory including current sub majors and user path.

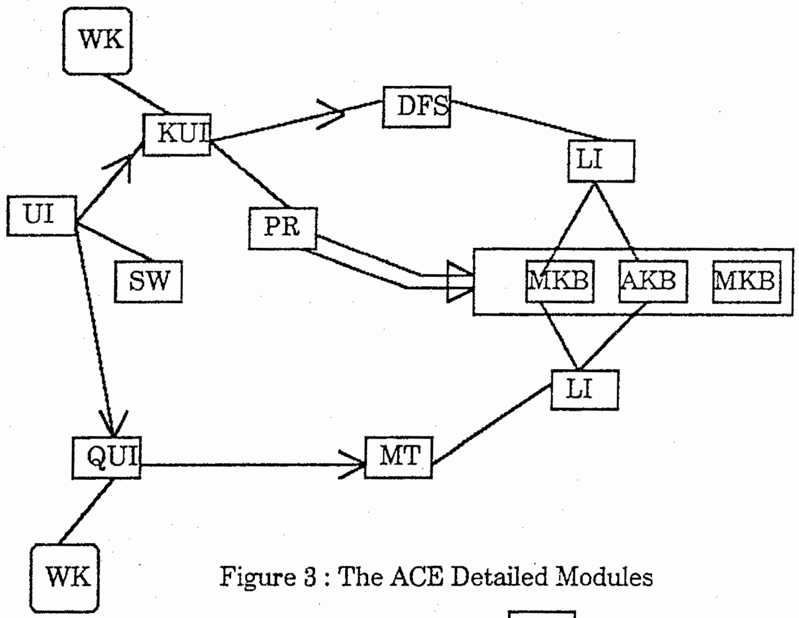
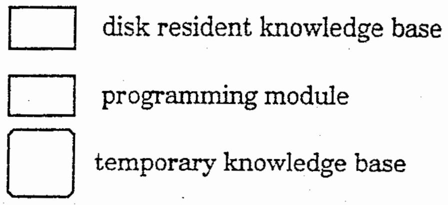


Figure 3 : The ACE Detailed Modules



3.1 The User Interface

ACE offers two modes of user interface. The first mode is the keyword user interface (KUI) and it allows the user to input the keywords (one or more) of the research area in which he or she is interested. There is a help facility to display the available keywords in KKB (PR). PR is a multi option command, with option "-k" it prints all available keywords in KKB. PR may also be issued with a sub-major as an argument to display its associated keywords. Table 1 shows PR command options and arguments. The first argument is mandatory while the second is optional.

Table 1: PR Options and Arguments

argument1	argument_2	function
-m		print all majors (all major tree)
-m	"major"	print its associated sub majors
-k		print all keywords
-k	"major"	print the major associated keywords
-a		print all advisors
-a	"major"	print the major associated advisors
"major"		check if the given major is in MKB
"keyword"		check if the given keyword is in KKB
"advisor"		check if the given advisor is in AKB

"major", "keyword", and "advisor" are replaced by the user input and the others are submitted as it is

The major may be submitted as a full path name major name starting from the major tree root. ACE denies a keyword if it is not from the stored keywords. If the submitted keywords are associated with more than one major, then ACE displays all the associated majors so the user can be more specific by submitting more keywords.

The second mode is the query user interface (QUI) and through this interface mode the user consults ACE using a simple query language. The ACE displays the majors from the MKB (initially starting from the major tree root). If the user selects one of the displayed majors, then ACE responds by displaying the sub-majors of the selected major (if any). The user may again select one of the resultant sub-majors to have sub majors of sub majors and so on. So, the user can go deeper and deeper into the major tree. If there are no sub-majors under the selected major, ACE notifies the user and displays the advisors list attached with the selected sub major node. Instead of selecting from the displayed sub majors, the user may input list command (LI) then ACE responds by displaying the advisors attached with the last selected major (current major). The user may terminate the dialogue by the exit command (EX) to let the system returns to the main menu module UI (choosing the interface mode: QUI or KUI).

The system offers the user the ability to navigate through the major tree by using the CD command (the same syntax of cd UNIX or DOS command). The user may turn back to the parent node (major) of the displayed child nodes (displayed submajors by typing "CD .." without quotes. A certain major may be located by specifying the major node full path as an argument with format as "CD path". The full path starts from the major tree root and states all nodes down to the desired node separated by/. Locating a certain node results in displaying its submajors and advisors. For example the submajors: distributed database design (dn), concurrency control (cc), query processing (qp), file allocation (fa) are displayed assuming that distributed database (ddb) is the current major (ACE is now positioned at ddb node). Instead of selecting one of them, the user enters "CD . ."

letting the system goes one level up (db node) and displays db submajors: sm, ndb, ddb, and rdb. The user now can inspect another major of the new displayed list. Table 2 summarizes the QUI language commands.

Table 2: QUI Command Summary

command	Function
CD/	locate the tree root displaying the root ubmajors
CD	path locate the given node displaying its submajors
CD..	locate the grand parent of the displayed submajors (sub) majr display submajors of the given (sub)major.
LI	list the advisors attached with the current node (parent of the displayed submajors)
EX	return to the main menenu

3.2 The ACE Inference Engine

The ACE inference engine design is dependent on the interface mode. Through the keyword interface the user is allowed to input one or more keywords of the research area in which he or she is interested. ACE tries to match the submitted keywords with the keywords of each major. If the match is happened, all majors with matched keywords are displayed. Samples of the production rules to control the matching process are given below.

Rule 1

```
IF      the keywords of the research area and the
        keywords of the current major are exclusive
THEN    backtrack to the parent of the current major
        and reactivate Rule 1
```

Rule 2

```
IF      the keywords of the research area are subset
        of the keywords of the current major
THEN    display the advisors associated with the
        current major select new major as a child
        of the current major to be the new current
        continue message to the user
```

Rule 3

```
IF      affirmative response from user to the continue
        message sign results from Rule 2
THEN    reactivate Rule 2

ELSE    exit
```

If the submitted keywords are found to be a subset of a certain major, the associated advisors are listed and the user is asked if he or she wants to continue (be more specific). If the user responds yes the inference engine continues to check the keywords of other majors (attempting children before backtracking), otherwise the engine stops.

In the query interface mode ACE displays the majors one at a time starting from the root of the major tree. As has been discussed before, the user can select one of the displayed majors then ACE responds by displaying the submajors of the selected major (if any). So the user can go deeper into the major tree checking the advisor list associated with the current major.

4. CONCLUSION

This paper presented the advisor expert system ACE which advises a graduate student to select research advisor(s) with interests close to the student research area. The implemented knowledge base is based on computer science majors, but it may be replaced with other department majors. The majors are represented as a tree and the system offers the user the ability to navigate through the tree. ACE has two interface types facilitating the user and the system interaction. The ACE power may be extended by allowing natural language interface.

REFERENCES

1. D, S, Nau, "Expert Computer Systems," IEEE Computers, Feb. 1983, pp. 63-85.

2. Donald A. Waterman, "A Guide to Expert Systems", Addison-Wesley, 1986.

3. Turbo Prolog Owner's Handbook, Borland, CA, USA, 1986.

4. L. Sterling, E. Shapiro, "The Art of Prolog: Advanced Programming Technique,"The MIT Press, Oct. 1986.

5. F. Roth, "The Knowledge-Based Expert System: Tutorial," IEEE Computers, Sep. 1984.

6. A. Barr et al, "Handbook of Artificial Intelligence," Heuristech Press, Vol. 2, 1982,pp. 229-235.

7. A. Ghonamei, M. Habashi, "An Expert System for Computer Configuration," Ain-Shams University Conference for Statistics Computation, March 1988.

Appendix A
Major Tree Abbreviations

cs	computer science	ds	distributed systems
sm	systems	sw	software
hw	hardware	ai	artificial intelligence
es	expert systems	ed	educational systems
mp	mineral explorations	vs	vision
rb	robotics	nl	natural language
sp	systems programming	os	operating systems
mm	memory management	fm	file management
ld	loader	is	information systems
dp	database	sa	system analysis
qb	query processing	cc	concurrency control
fa	file allocation	sd	semantic modeling
rd	relational databases	hd	hierarchical databases
nd	network databases	cp	compilers
al	algorithms	op	optimization
fl	formal languages	lb	load balance
ct	theory of computation	vp	visual programming
cm	control systems	lp	logic programming
dn	distributed database design		
ddb	distributed database		
dos	distributed operating system		

نظام خبير لتحديد مشرف الماده العلميه

النظام الخبير فى البحث يساعد طالب الدراسات العليا فى تحديد مشرف او مشرفين لرسالته العلميه فى نقطة البحث التى يريد الطالب التسجيل فيها بحيث تكون اهتمامات قائمة المشرفين المنتقاه هى الاقرب لنقطة البحث. واهتمامات المشرف يراعى فيها التنوع والتخصص المتدرج.