Expert System: The New Approach of Construction Drawn from Comparison of Different Expert System.

Four Examples of Knowledge Engineering Languages

Chandrajeet Borkar[1].
Final Year Information Technology Department
Email Address
chandrajeetborkar@gmail.com
Phone number: 9823444021
College
G. H. RAISONI COLLEGE OF ENGINEERING AND MANAGEMENT, AMRAVATI, MAHARASTHRA INDIA

Abstract— An expert system is a computer program that exhibits high performance in a specific problem domain due to a large amount or formally encoded knowledge and the ability to conduct formal reasoning on this knowledge. An expert system is designed to do various tasks that an expert would typically perform: diagnose, interpret, Consult, classify, identify, Search through a space or possible solutions, explain, tutor, and analyze. In expert systems, domain knowledge is often represented as a set of production rules.
This paper deals with 4 knowledge based engineering languages and there examples which are mention here in the paper. Examples of Expert system mention are still in use by many of organization. Four knowledge engineering languages merit our special attention because they are so widely used. Also Paper Conclude With some of improvement required in Expert system and thus it could again spin the stop wheel of ES.

Index Terms: Expert System, Knowledge, Artificial Intelligence, Knowledge base, KB Engineering Languages, EMYCIN, EXPERT, ADVICE, and ROSIE.

I. INTRODUCTION

An Expert System is a computer program that has the capability of decision-making ability of a human expert. In expert systems, domain knowledge is often represented as a set of production rules. These rules take the form of: IF <condition> THEN <action>. In many expert systems, some form or probable or plausible reasoning is used in the inference procedure to handle uncertainties. Data is often represented as value/confidence pairs. In addition, an expert system includes the memory needed to store intermediate results of rules when they are actuated or fired. A favorite starting point for expert system architecture is to organize all three components, i.e., knowledge base, rule evaluator, and User Interface.

At present, the technology of expert systems is undergoing very rapid fall and by understanding the modern terms, it's possible of being applied to overcome a wide spectrum of practical problems. Current systems, however, suffer from a number of limitations that restrict their usefulness. They typically employ only one form of knowledge representation, usually a rule-base representation. They have no learning abilities, use only one type of inference procedure, and use only a single control strategy.

Our research on the five knowledge base languages to built an Expert System includes EMYCIN, EXPERT, ADVISE and ROSIE system has the goal of overcoming these limitations while comparison is been done in between all Expert System which are included in paper. This paper concludes with some of the suggestions to overcome the problem that have made the expert system so less popular in this modern world.

Table 1. Knowledge Engineering Languages for Building expert Systems.

Tool	**Features**	**Implementable languages and Developer**
EMYCIN	-Rule based -Backward chaining -Certainty handling -Explanation -Acquisition	INTERLISP Stanford University
EXPERT	-Rule-based -Forward chaining -Certainty handling -Explanation	FORTRAN Rutgers

	-Acquisition -Consistency checking	University
ADVICE	-Rule-based -Mixed chaining -Certainty -Control Strategies -Control Information	PASCAL University of Illinois
ROSIE	-Rule-based -Forward chaining -Procedure-oriented -English-like syntax	INTERLISP The Rand Corporation

Here is the detail description which shows all the comparison between all the tools used and the different expert system which a small rules which are and are still used. Please make a note this are much classified rules and are bared with user they don't bear any relation with paper presentation.

II. EMYCIN

EMYCIN uses rule-based knowledge representation scheme with a rigid backward chaining control mechanism that limits its application to diagnosis and classification-type problems. However, the system provides sophisticated explanation and acquisition facilities that clearly speed expert system development.

An EMYCIN rules has the form IF antecedent THEN consequent, where the antecedent is a collection of true/false expression and the consequent is a conclusion that follows the antecedent. A context tree organized EMYCIN objects in a simple hierarchy and provides some of the inheritance characteristics of a frame system. EMYCIN associates a certainty values ranging from -1(false) to +1(true) with every expression in the antecedent. The IF portion of the rule is considered to be true if its certainty is greater than some threshold and false if below some other threshold. EMYCIN uses special evidence-combining formulas to decide how to combine the certainties in the antecedent and update the certainty of the consequent.

An EMYCIN rule from SACON expert system.

English Translated Rule from EMYCIN:

If :

1) The material composing the substructure is one of: the metals, and

2) The analysis error (in percent) that is tolerable is between 5 and 30, and

3) The non-dimensional stress of the substructure is greater than 0.9, and

4) The number of cycles the loading is to be applied is between 1000 and 10000

Then

It is definite (1.0) that fatigue is one of the stress behavior phenomena in the substructure.

Actual EMYCIN rule:

PREMISE:
($AND (SAME CNTXT MATERIAL (LISTOFMETALS))
(BETWEEN* CNTXT ERROR 5 30)
(GREATERP* CNTXT NO-STRESS 0.9)
(BETWEEN* CNTXT CYCLE 1000 100000))
ACTION:
(CONCLUDE CNTXT SS-STRESS FATIGUE TALLY 1.0)

The above Shown is a rule from SACON, a consultation system that provides advice to a structural engineer regarding the use of a structural analysis program called MARC. MARC uses mathematical analysis techniques to simulate the mechanical behaviors of objects.

III. EXPERT

EXPERT uses a rule-based knowledge representation scheme and had a limited forward chaining control mechanism that makes it suitable for diagnosis and classification-type problem. EXPERT has built-in explanation, knowledge acquisition, and consistency checking module works by storing a data base of representative cases with known conclusion and using it to test the expert system after the knowledge engineer adds rules. If a case doesn't produce correct conclusion, EXPERT displays the reasoning for that case so that the knowledge engineer can understand how the new rules led to the unexpected results.

EXPERT was designed to handle consultation problem in medicine, it structures knowledge to facilitate medical interpretation. Rules in EXPERT distinguish between finding and hypotheses. Findings are observations like a patient's age or blood pressure, while hypotheses are conclusion inferred from finding or other hypotheses. In EXPERT, finding have a form f (finding-name, certainty-interval), while hypotheses have the form h (hypothesis-name, certainty-interval). The truth value is t if the finding is true and f is false. The certainty interval represents the confidence the expert has in the hypothesis, e.g. h (matrl,0.2:1) means conclude hypothesis material with the confidence of 0.2 to 1. Confidence values range from -1(complete denial) to 1(complete confirmation).

An EXPERT rule from the AI/RHEUM expert system

EXPERT Rule:

**hypotheses

CNC the patient has a central nervous system disease

**finding

SEIZ seizures occur
PHYCH psychosis exists

OBSYN organic brain syndrome is present
COMA coma exists

English translation of the EXPERT rule
**rule
IF:
One of the following is true:
Seizures, psychosis, organic brain syndrome, or coma
THEN:
Conclude serious central nervous system disease
At a confidence level of 1.0.

Actual EXPERT RULE
[1: f(seiz,t), f(psych,t), f(obsyn,t),f(coma,t) -> h(cnc,1.0)].

IV. ADVICE

ADVICE is general-purpose, System-building aid consisting of an integrated set of development tools. the tools include support for multiple forms of knowledge representation(rules, semantic net and relational data base), support for several certainty propagation scheme(probabilistic, approximate ,min/max logic and weighting evidence), support various control strategies(utility optimization, probabilistic network traversal, and forward and backward rule chaining), and the incorporation of inductive learning programs(GEM and CLUSTER) for inductively deriving decision rules and control information from examples. In explanation made, ADVICE paraphrases decision rules, allow simple interrogation of knowledge base, and display its reasoning steps. ADVICE is implemented in PASCAL and operates under UNIX operating system. ADVICE was developed by University of Illinois. ADVISE on user and developer flexibility and convenience. There is the design goal of getting ADVISE out to the public. This is a major reason why Pascal was chosen as the implementation language. Good versions of Pascal are available on many microcomputer systems. There are two methods to support microcomputers. One method is hand tailoring ADVISE to the microcomputer environment. The other more attractive option is automatic tailoring. A tool for ADVISE to do automatic tailoring is being developed.

ADVISE Architecture in short: The four major components or the ADVISE system are: (1) Control Block and User (2) Interface Knowledge Base (3) Query Block (4) Knowledge Acquisition Block
Rule base: The rule base contains rules in the basic form: CONDITION ::> CONCLUSION: α,β
Where

CONDITION is a formal expression in GVL l variable-valued logic which involves elementary conditional statements (called selectors), linked by various logic operators (including quantifiers).

CONCLUSION defines the decision or action which is to be taken when the CONDITION is satisfied by a given situation.

α is the strength or evidence which supports CONCLUSION when the CONDITION is completely satisfied ($0 \le \alpha \le 1$).

β is the strength of evidence which supports the negation of CONCLUSION when the CONDITION is not satisfied ($0 \le \beta \le 1$).

An ADVICE rule from PLANT/CD expert system.

Below is the English form of an example rule from PLANT/CD:

RULE12
[This rule is tried in order to find out about Black Cutworm development]
IF:
1) The degree-day table is known,
2) The observation date < planting date, and
3) The egg population in the field is known.
Then:
Simulate BCW development and assign the results to the variable BCWD and display results of the simulation.

Actual PLANT/CD rule: This rule is formulated in PLANT/CD as:
RULE12
[DD = KNOWN]OBDATE < PLANTDATE][FEGGS = KNOWN] ➔
[BCWD = TRAP (15, OBDATE, PLANTDATE, FEGGS, DD)]TRAP (9, OBDATE)
! SIMULATE BCW DEVELOPMENT AND DISPLAY RESULTS.

Where DD denotes degree-day table, OBDATE denotes observation date, PLANTDATE denotes planting date, FEGGS denotes the egg population in the field, BCWD denotes the results of BCW development,
TRAP(15,OBDATE,PLANTDATE,FEGGS,DD) is a function to simulate BCW Development and is used to assign a value to BCWD, TRAP(9,OBDATE) is a function that displays the results of BCW development and is treated as a predicate by the ADVISE RULE PARSER, and 15 and 9 indicate what TRAP functions to call.

V. ROSIE

This general-purpose knowledge engineering language combines a rule-based representation scheme with a procedure-oriented language design. Thus ROSIE programs are typically nested

procedures and functions, each defined as a set of rules. ROSIE has a English liked syntax that makes its code quite readable, powerful pattern matching routines for matching the rules premises against the data, and control over remote jobs via interface to the local operating system. ROSIE's supports environment include editing and debugging tools but no built-in expression or acquisition facilities.

ROSIE has been used to build expert systems in variety of problem domain, including law, crisis management and the military.

Programs take the form of rule sets, each defined to be a procedure, a generator, or a predicate. A procedure is like a subroutine: performs some task and then returns control to the portion of the program that called it. A generator is like a function: it returns a value or set of values. For example, a generator for determining medical costs would return a specific dollar amount when given the name of an injured party. A predicate is a function that always returns either true or false. For example, LDS has a predicate that decide whether or not the product is defective.

The example given below is an actual ROSIE rule from LDS, an expert system for analyzing product liability cases. The system uses the facts of the case, together with rules based on formal legal principle and attorney's informal procedures and strategies, to calculate defendant liability, case worth, and an equitable settlement amount.

The two ROSIE rules in below represent executable code, not English translation of the code. ROSIE's expressiveness and readability expert system development, especially in the domains where the rules are naturally complex and detailed.

Two ROSIE rules from LDS expert system

Actual ROSIE rule

If

There is a test for product inspection and
That test is recognized by the experts as good and sound and
That is used for possible discovery of defects and
The defendant did perform that test

Assert

The product was defective for failure to test and inspect.

Actual ROSIE rule

If

The product was dangerous to a substantial number of people and
The plaintiff was injured by the product and
The product is represented by the defendant and
(The defendant did not warn of the danger or
The warning was not complete or
The warning was insufficient) and
The normal use of the product was both intended and foreseeable.

Assert

The product was defective for failure to warm.

VI. Authors Point of View for Construction of ES

Today, In world the wheel of expert system is been slowed to a great extend because of the limitation those occurs while working with Expert System. Main Limitation can be enlisted which could draw main attention while working with an Expert System, are Slow Execution, Support facilities, execution related to environment, problem domain and structure the shortcuts of Expert in Knowledge based, including the Extraction of that knowledge for the Bases in ES and List is no finish on this problems only. Hence while studying and considering the above paper, Knowledge Engineer or constructors of Expert System could come to a conclusion which should be consider while implementation of know and construction of ES. Because of below points mention Expert System could be enhance and knowledge engineer could limit some of the limitation offer while construction and after construction either. Here Author, Suggest some of consideration while Building an Expert system or Knowledge base. Those points are enlisted below:

- To increases the speed of execution, the Inference engine should be kept behind the construction (i.e. at the end part of output) or should be excluded. More efficient use of concept such as state switching algorithm or Data Structure more efficient should be use. Those, Data Structure could be Tress, Graphs, Link List or pattern matching or recognition.
- As its said, algorithm are ultimate solution of the structured data but processing on the Un-structurized Data could be possible by using Expert System. This should also be consider that expert system output is always and data which could be process future more. This Expert System output is a conclusion which just used to observe and take decision. This conclusion could be feed to automation, which is in need of future more commands. This Automation may be waiting for human conclusion for proper operation and if expert system replace's human, Hence Automation could gain Conclusion more precise in nature from Expert System. The conclusion of Expert System could handle process and give commands depending upon its output. Hence achieving more Automation.
- It's observed from studying of various papers that various knowledge bases are implemented in

different domain and more of all in different implemented languages which varies in great phase's or extend. The knowledge engineer and team should restrict themselves to only one implementable language and only one knowledge base domain. A specific standard should be defined for construction of an Expert system which will define restriction on Knowledge base (Database) language domain and implementation language should be restricted to only one.

- By achieving the above restriction or standard, it's possible to support and from a chain of expert system. Now be above restriction we could feed an output of one Expert System to other which may be in need of manual feeding of data. That is Output of an Expert system could become input to another expert system and hence more automation could be achieved.
- Common Sense of one domain is also achieve by stucturizing the data into a knowledge base but restricted to only one Domain. This could make more advancement in automation and also will turn Robotics to a Golden era of automation.
- Only one typical employ form of knowledge representation, usually a rule-base representation limits the usefulness of expert system. There are various control strategies either present to overcome this drawback.

Thus it can conclude that by using Expert System in true senses, Entity related to automation could achieve more automation in the field of computer science & Application. There are various development still required and a very modern term to become Definition of the Expert system. The wheel of Expert system development is stop somewhere but if users understand that valuable contribution of the Expert system in true sense and make modification according to authors view, it could be assure the history of Automation would travel and start to a Golden Pages.

REFERENCES

Papers:

1. *A case study in applying the general purpose inference system ADVICE to predicting black cutworm damage in corn, M.S. thesis, computer science dept. University of Illinois at champage-urbana,july 1983(PLANT/CD).*
2. *Van Melle, W., Shortliffe, E. H.., and Buchana, B. G. EMYCIN: A Domain-Independent system that Aids in constructing knowledge-based consultant programs. Machine Intelligence, InfotechState of the Art report 9, no. 3, 1981.*
3. *Lindberg, D. A. B., Sharp, G. C., Kingsland L.C., Weiss, S.M. Hages, S.P. Ueno, H., and Hazelwood, S.E. Computer based rheumatology consultant. Proceedings to the Third world conference on the medical informatics, 1980.*
4. *Lindberg, D. A. B., Gaston, L.W., Kingsland, L.C., and Vanker, A. D. A1/COAG, a knowledge-based system for consultation about human hemortasis disorders: progress report. Proceeding of the fifth annual symposium on computer application in medical care, 1981.*
5. *MARC User Information Manual available from MARC analysis Research corporation, Palo Alto, Ca, September, 1976.*
6. *Meldman, J.A., A preliminary Study in computer-aided legal analysis, PhD thesis, Departments of Electrical Engineering and computer science, MIT, August, 1975.*
7. *Barstow, D. R., Aiello; N., Duda, R.O., Erman, L. D. Forgy, C., Gorlin, D., Greiner, R. D., Lenat , D. B., London, P.t., McDermott, J., Nii, H. P., Politakis, P., Reboh, R. Rosenschein, S., Scott A.C., Van Melle, W., and Weiss, S.M. Languages and tools for Knowledge Engineering. In F. Hayes.-Roth, D. A. Waterman, and D. B. Lenet(eds.) Building Expert Systems, Reading, Mass.: Addison-Wesley,1983.*
8. *Waterman, D. A. and Peterson, M., Rule-based models of Legal expertise. Proceedings of the First Annual National Conference on Artificial Intelligence, 1980*
9. *Waterman, D. A. and Peterson, M. Models of Legal Decision-making. Rand Report R-2717-ICJ, The Rand Corporation, 1981.*
10. *S. J. Rannaugh, M., Kastner, J. K., Schor, R; and Van Woerkom, H. M. YES/MVS: A Continues real time Expert System. Proceedings AAAI-84, 1984.*
11. *Schor, M. I. Using Declaratives knowledge representation techniques: implementing truth maintaine in OP5. Proceeding of the First conference on Artificial Intelligence Application, IEEE Computer Society, December 1984.*
12. *Fain, J., Gorlin, D., Hayes-Roth, F., Rosenschein, S., Sowizral, H., and Waterman, D. A. The ROSIC Language Reference manual N-1647-ARPA, The Rand Corporation, 1981.*
13. *Forgy, C. L. Rete: Overview of OP5. Computer Science Dept. report. Carnegie-Mellon University, Pittsburgh, Pa., June 1983.*