Impact Factor Value 4.046 **e-ISSN: 2456-3463**
National Conference on "Recent Advances in Engineering and Technology" SAMMANTRANA 19
Organized by Government College of Engineering, Nagpur
International Journal of Innovations in Engineering and Science, Vol 4 No.8, 2019
www.ijies.net

"Expert System: The New Approach of Construction drawn from Comparison of Different Expert System"

Four Examples of Knowledge Engineering Languages

Chandrajeet Borkar

Instructor, Computer Science & Engineering Department
Government College Of Engineering, Nagpur, Maharashtra 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 space or possible solutions, explain, tutor, and analyze.*
In expert systems, domain knowledge is often represented as a set of production rules. These rules take the form of:
IF <condition> THEN <action>
This paper deals with 4 knowledge-based engineering languages and their 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 the improvement required in the Expert system and thus it could again spin the stop wheel of ES.

Keywords*: Expert System, Knowledge, Artificial Intelligence, Knowledgebase, KB Engineering Languages.*

I. INTRODUCTION

Expert System In artificial intelligence, an expert system is a computer system that emulates the decision-making ability of a human expert. Expert systems are designed to solve complex problems by reasoning about knowledge, like an expert, and not by following the procedure of a developer as is the case in conventional programming. The first expert systems were created in the 1970s and then proliferated in the 1980s. Expert systems were among the first truly successful forms of AI software.

An expert system has a unique structure, different from traditional programs. It is divided into two parts, one fixed, independent of the expert system: the inference engine, and one variable: the knowledge base. To run an expert system, the engine reasons about the knowledge base like a human. In the '80s a third part appeared: a dialog interface to communicate with users. This ability to conduct a conversation with users was later called "conversational".

II. Software architecture

The rule base or knowledge base

In expert system technology, the knowledge base is expressed with natural language rules IF ... THEN ... For examples:

- "IF it is living THEN it is mortal"
- "IF his age = known THEN his year of birth = date of today - his age in years"
- "*IF the identity of the germ is not known with certainty AND the germ is gram-positive AND the morphology of the organism is "rod" AND the germ is aerobic THEN there is a strong*

probability (0.8) that the germ is of type Enterobacteriaceae".

This formulation has the advantage of speaking in an everyday language which is very rare in computer science (a classic program is coded). Rules express the knowledge to be exploited by the expert system. There exist other formulations of rules, which are not in everyday language, understandable only to computer scientists. Each rule style is adapted to an engine style. The whole problem of expert systems is to collect this knowledge, usually unconscious, from the experts. There are methods but almost all are usable only by computer scientists.

III. The inference engine

The inference engine is a computer program designed to produce reasoning on rules. In order to produce reasoning, it is based on logic. There are several kinds of logic: propositional logic, predicates of order 1 or more, epistemic logic, modal logic, temporal logic, fuzzy logic, etc. Except for propositional logic, all are complex and can only be understood by mathematicians, logicians or computer scientists. Propositional logic is the basic human logic, which expressed in the syllogism. The expert system that uses that logic is also called a zeroth-order expert system.

To guide a dialogue, the engine may have several levels of sophistication: "forward chaining", "backward chaining" and "mixed chaining". Forward chaining is the questioning of an expert who has no idea of the solution and investigates progressively (e.g. fault diagnosis). In backward chaining, the engine has an idea of the target (e.g. is it okay or not? Or: there is a danger but what is the level?). It starts from the goal in hopes of finding the solution as soon as possible. In mixed chaining, the engine has an idea of the goal but it is not enough: it deduces in forward chaining from previous user responses all that is possible before asking the next question. So, quite often, he deduces the answer to the next question before asking it.

IV. Knowledge engineering

The building, maintaining, and development of expert systems is known as *knowledge engineering*. Knowledge engineering is a "discipline that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise".

There are generally three individuals having interaction in an expert system. Primary among these is the end-user, the individual who uses the system for its problem-solving assistance. In the construction and maintenance of the system there are two other roles: the problem domain expert who builds the system and supplies the knowledge base, and a knowledge engineer who assists the experts in determining the representation of their knowledge, enters this knowledge into an explanation module and who defines the inference technique required to solve the problem. Usually, the knowledge engineer will represent the problem-solving activity in the form of rules. When these rules are created from domain expertise, the knowledge base stores the rules of the expert system.

Table 1. Knowledge of 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 Acquisition Consistency checking	FORTRAN Rutgers University
OPS5	Rule-based Forward chaining Flexible control Flexible representation	FRANZ LISP Carnegie-Mellon University

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 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 the paper presentation.

V. EMYCIN

This skeletal knowledge engineering language is essentially MYCIN with the domain knowledge removed. EMYCIN uses a 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 rule 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 value 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 (say0.2) and false if below some other threshold say (-0.2). 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 the SACON expert system.

EMYCIN Rule:

- 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.

(English translation of the EMYCIN rule shown below)

Actual EMYCIN rule

- PREMISE:

($AND (SAME CNTXT MATERIAL (LISTOFMETALS))

(BETWEEN* CNTXT ERROR 5 30)

(GREATER* 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.

VI. EXPERT

This skeletal knowledge engineering language 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 the built-in explanation, knowledge acquisition, and consistency checking module works by storing a database of representative cases with a known conclusion and using it to test the expert system after the knowledge engineer adds rules. If a case doesn't produce the correct conclusion, the EXPERT displays the reasoning for that case so that the knowledge engineer can understand how the new rules led to unexpected results.

EXPERT has been used to build diagnosis programs in medicine, geology, and other areas. Since the EXPERT was designed to handle a 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 has 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

**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

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

VII. OPS5

This general-propose knowledge engineering language used a rule-based representation scheme that works via forward chaining. The system's generality supports diverse data representation and control structures within the single program. OPS5 has a powerful pattern-matcher and an efficient interpreter for matching rules against the data but lacks a sophisticated supports environment. It has no built-in explanation or acquisition mechanisms and only minimal facilities for program editing and debugging. OPS5 is the latest in a succession of similar rule-based languages (e.g., OPS, OPS4) that evolved from work at Carnegie-Mellon Uni9versity in developing programming languages for modeling human cognition and memory.

OPS5 and the earlier languages in the OPS5 series have been used for many cognitive psychologies, AI and expert system application.

An OPS5 rule has the from antecedent → consequent, where the antecedent describe data element and the consequent specifies the actions to take if the antecedent matches the database. Data elements in OPS5 are objects described by a set of attribute-value pairs. They look a bit LISP expression, as illustrated

In English:

The tall woman is 23 years old.

In OPS5:

(WOMAN ↑HEIGHT TALL ↑AGE 23)

The object (e.g. WOMAN) comes first followed by the attribute-value pairs. Attributes are marks with a caret (↑) to distinguish them from values. One thing that makes OPS5 (and LISP) so difficult to read is the use of one-word tears to stand for complex concepts.

An OPS5 rule from the YES/MVS expert system

English Translation of OPS5 rule

- IF: The current task is to maintain the job entry system queue space and the queue space is critically low and there is a link to the computer that is actively receiving a message

- THEN: Send a command to cut the link and mark the link's reception status as "about to be NO"

Actual OPS5 rule

```
(P STOP-RECEPTION
(TAKS ↑ TASK-ID JES-Q-SPACE)
(JES-Q ↑ MODE PANIC)
(<THE LINK> (LINK ↑ID<L-ID>
                  ↑ STATUS <<ACTIVE I/O-
ACTIVE>>
                  ↑ RECEIVE YES))
→
(CALL REMOTE-MAKE
LINK-COMMAND      ↑ID<L-ID>
                  ↑RECEIVE NO
                  ↑RM-TO: MCCF
(MODIFY <THE-LIN> ↑RECEIVE TO-BE-NO))
```

OPS5 rules can be somewhat verbose and unintelligible. Even worse, the OPS5 language has no provision for displaying English versions of the rules to the user the way EMYCIN does. Despite this, OPS5 is one of the most widely used knowledge engineering languages available, and it has been widespread use is due partly to its execution efficiency and partly to its ready availability.

VIII. ROSIE

This general-purpose knowledge engineering language combines a rule-based representation scheme with procedure-oriented language design. Thus ROSIE programs are typically nested procedures and functions, each defined as a set of rules. ROSIE has an English liked syntax that makes its code quite readable, powerful pattern matching routines for matching the premises of the rules against the data, and control over remote jobs via an interface to the local operating system. ROSIE's supports environment includes editing and debugging tools but no built-in expression or acquisition facilities.

ROSIE has been used to built expert systems in a 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 decides 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 below represent executable code, not the 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 the 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.

IX. Authors Point of View for Construction of ES

Today in the world the wheel of an expert system is been slowed to a great extent. Rather Author says that it's been totally stopped because of the Execution time taken by the Expert System and a lot of factors in similar ways. Here I, Suggest some of Building an Expert system. Those points are enlisted below

- While Construction of an expert system Knowledge engineering should restrict himself and all another knowledge engineer to one Implement Language only. That is an expert system knowledge-based (Clearly Data Base) should be designed in a single and high-speed Extractable Knowledge Base and also should include a various feature by itself.
- To increase the speed of the Expert system we should exclude Inference Engine and to achieve a great speed, we could use the Concept of state switching and the Data Structures like Graphs and Pattern matching algorithms.
- If expert system is been resisted to one domain implementable language then one Expert system could support another expert system and could also support the chain of Expert system and hence we could solve the multiple domain problems from the result of one expert system to another.
- The result of Expert system that is the output of the expert system are the various time a data which is Structured and could be feed to an algorithm and hence more future be operation on the data could be done as in more sophisticated manner

Thus I can conclude that by use Expert System in true senses we could achieve more automation in the field of Expert System. 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 stopped somewhere but if you understand that valuable contribution of the Expert system in the true sense and make modification according to my view, I could assure the history of Automation would travel and start to a Golden Pages.

REFERENCE

[1] Donald Waterman (1986), A guide to Expert Systems, Published by Dorling Kindersley (India), pvt ltd, Licensees of Pearson Education in South Asia

[2] Dan W. Patterson (1990) , Introduction to Artificial Intelligence and Expert System, Published by Pearson Education Inc., Publishing as Prentics Hall.

[3] Stuart Russell & Peter Norvig (), Artificial Intelligence A Modern Approach (2 edition), published by Dorling Kindersley (India) Pvt. Ltd, Licensees of Pearson Education in South Asia,

[4] Seymour Lipsuchutz (2006), Data Structures, Published by Tata McGraw Hill Education Private Limited.

[5] Jackson, Peter (1998), Introduction To Expert Systems (3 ed.), Addison Wesley, p. 2.

[6] The Expert System Plant/Cd: A Case Study In Applying The General Purpose Inference System Advise To Predicting Black Cutworm Damage In Corn By Albert Gerard Boulanger B.S., University or Florida., 1978 THESIS Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University, Ulinois at Urbana-Champaign ,Urbana, Illinois.