Expert Systems

TABLE OF CONTENTS

1. Introduction

Human experts are able to perform at a high level because they know great deal about their areas of expertise. This simple observation is the underlying rationale for the design of knowledge-based problem solvers. An expert system, as these programs are often called, uses domain specific knowledge to provide expert quality performance in a problem domain. Generally, expert system designers acquire this knowledge with the help of human domain experts, and the system emulates their methodology and performance. As with skilled humans, expert systems tend to be specialists, focusing on a narrow set of problems [3].

Expert systems neither copy the structure of the human mind, nor are they mechanisms for general intelligence. They are practical programs that use heuristic strategies developed by humans to solve specific classes of problems [3].

2. The Benefits of Expert Systems

The benefits of expert systems (ES) can be listed as follows:

- Increased output and productivity: As compared with humans, ES can work faster than humans, requiring fewer workers and reducing cost.
- Increased quality: ES can increase quality by providing consistent advice and reducing error rate.
- Reduced downtime: Using ES in diagnosing malfunctions and prescribing repairs, it is possible to reduce downtime significantly.
- Capture of scarce expertise
- Flexibility: In providing services and in manufacturing
- Easier equipment operation
- Elimination of the need for expensive equipment: In many cases a human must rely on expensive instruments for monitoring and control. ES can perform the same tasks with lower-cost instruments because of their ability to investigate more thoroughly and quickly the information provided by instruments.
- Operation in hazardous environments
- Accessibility to knowledge: ES make knowledge and information accessible to people.
- Reliability: ES are reliable in that they do not become tired or bored, and they consistently pay attention to all details and so do not overlook relevant information and potential solutions.
- Increased capabilities of other applications: Integration of ES with other systems makes the systems more effective; they cover more applications, work faster, and produce higher quality results
- Ability to work with incomplete and uncertain information

3. The Structure of an Expert System

A user interface simplifies communication with users and hides much of the system complexity [3].

The general knowledge base contains problem-solving knowledge of the particular application. In a rule-based expert system this knowledge is represented in the form of if… then… rules. Beside general knowledge, the knowledge base also contains case-specific information [3].

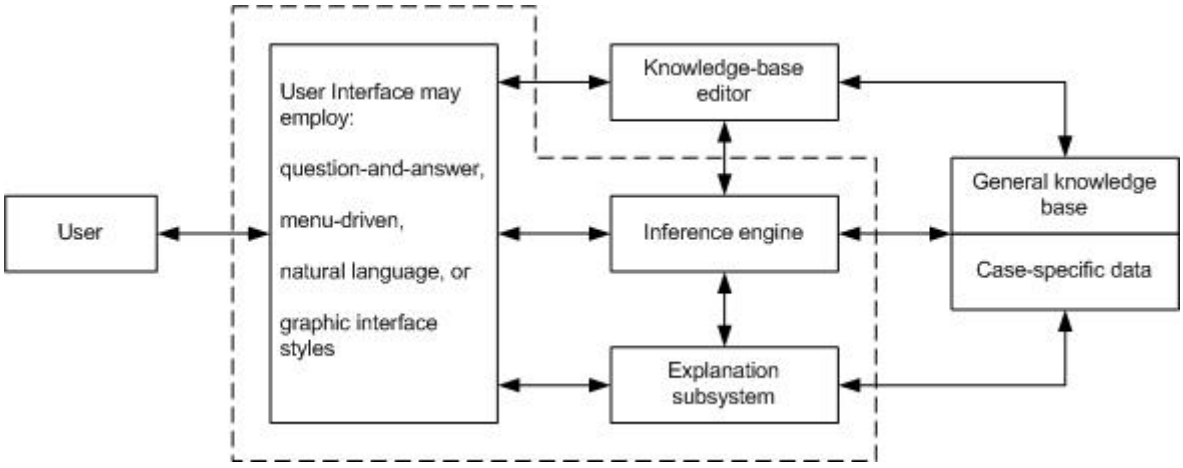

Figure 1: Architecture of a typical expert system [3]

The inference engine applies the knowledge to the solution of actual problems. It can be treated as an interpreter for the knowledge base. In the production system, the inference engine performs the recognize-act control cycle [3]. The steps of the recognize-act control cycle can be described as follows [2]:

1. Match the premise patterns of rules against elements in the working memory. The working memory contains a description of the current state of the world in a reasoning process [3].
2. If there is more than one rule that can be applied, choose one to apply in the conflict resolution. If no rule applicable, stop.
3. Apply the chosen rule, perhaps by adding a new item to the working memory or deleting an old one. If termination condition fulfilled stop, else go to step 1.

The termination condition is either defined by a goal state or by a cycle condition. For example, a cycle condition can be that the cycle will be executed as maximum 100 steps [2].

The case-specific data cover the facts, conclusions, and other information relevant to the case under consideration. This includes the data given in a problem instance, partial conclusions, confidence measures of conclusions, and dead ends in the search process. This information is separated from the general knowledge base [3].

The explanation subsystem allows the program to explain its reasoning to the user. These explanations include justifications for the system's conclusions, explanations of why the system needs a particular piece of data, and, where useful, tutorial explanations or deeper theoretical justifications of the program's actions [3].

Knowledge-base editors help the programmer locate and correct bugs in the program's performance, often accessing the information provided by the explanation subsystem. They also may assist in the addition of new knowledge, help maintain correct rule syntax, and perform consistency checks on the updated knowledge base [3].

4. The Production System

The production system provided a basis for modern expert system architecture. If a typical expert system is treated as a production system, the knowledge base is the set of production rules. The production rules are condition-action pairs and represented as if… then… rules.

IF
condition
THEN
action

If the condition of a rule is satisfied, the system asserts the rule's conclusion as true. Case-specific data are kept in the working memory. The inference engine implements the recognize-act cycle of the production system; this control may be either data-driven (forward) or goal-driven (backward).

5. Forward Chaining

Forward chaining or data-driven inference works from an initial state, and by looking at the premises of the rules (IF-part), perform the actions (THEN-part), possibly updating the knowledge base or working memory. This continues until no more rules can be applied or some cycle limit is met.

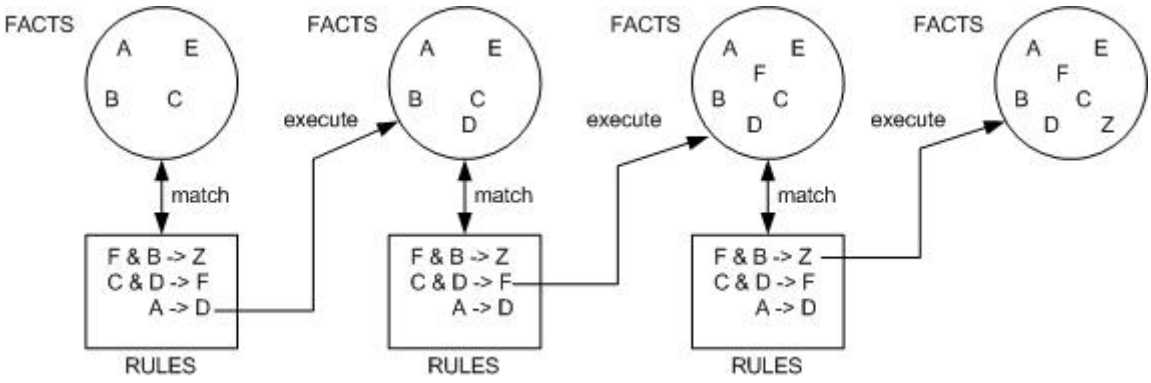

Figure 2: Forward chaining example

The problems with forward chaining can be summarized as follows:

- many rules may be applicable.
- the whole process is not directed towards a goal.

6. Backward Chaining

Backward-chaining inference engines start with a goal, or hypothesis, and work through the rules trying to match that goal with the action clauses (THEN part) of a rule. When a match is found, the condition clauses (IF part) of the matching rule become a "subgoal" and the cycle is repeated until a verifiable set of condition clauses is found. Backward-chaining rules are also called consequent rules.

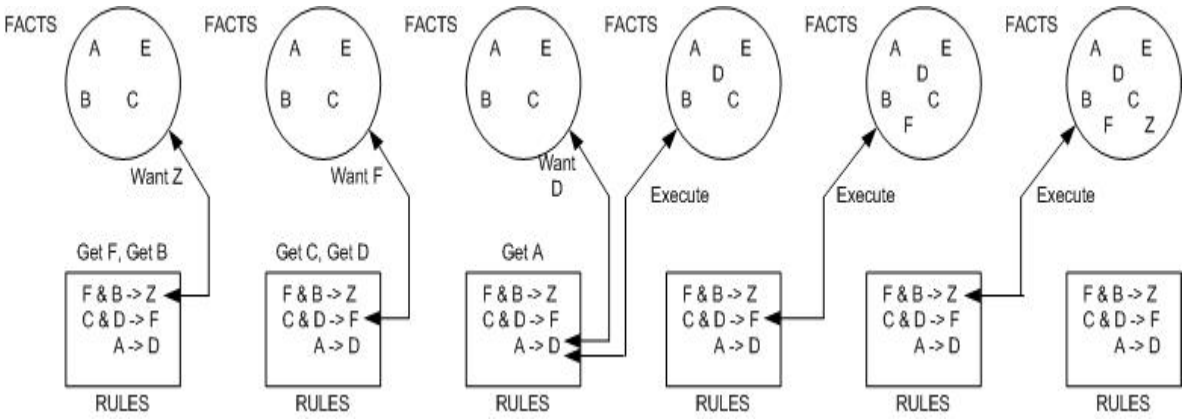

Figure 3: Backward chaining example

The advantage of backward chaining is that the search is directed, whereas its disadvantage of backward chaining is that the goal has to be known.

7. An Example of Goal-driven Problem Solving [3]

The rules for a small expert system for diagnosing automotive problems are given as follows:

Rule 1: **IF**
the engine is getting gas, and
the engine will turn over,
THEN
the problem is spark plugs.

Rule 2: **IF**
the engine does not turn over, and
the lights do not come on
THEN
the problem is battery or cables.

Rule 3: **IF**
the engine does not turn over, and

the lights do come on
THEN
the problem is the starter motor.

Rule 4: **IF**
there is gas in the fuel tank, and
there is gas in the carburetor
THEN
the engine is getting gas.

If the recognize-act control cycle is goal directed, the top-level goal is placed in working memory. In the example, the top level goal is "the problem is X" and is placed in working memory as shown figure 4. The variable X can match with any phrase such as the conclusion of the rule 2, "battery and cables".

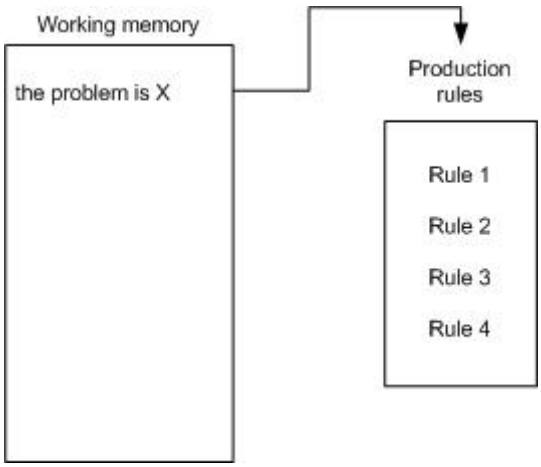

Figure 4: The production system at the start of a consultation in the car diagnostic example

In the example, three rules (1, 2, and 3) match with the expression in working memory. The decision of firing a rule is a matter of conflict resolution. This conflict can be resolved by firing the lowest-numbered rule, namely rule 1. This causes X to be bound to the value "spark plugs" and the premises of rule 1 to be placed in the working memory as in figure 5.

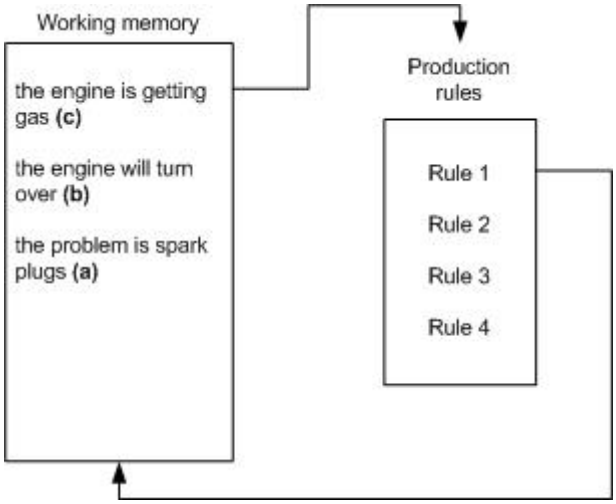

Figure 5: The production system after Rule 1 has fired

In order to prove that the conclusion of rule 1 (labeled with a in figure 5) is true, the two premises of rule 1 (labeled with b and c in figure 5) should be satisfied. The problem is divided into two subproblems in this state. Afterwards, the rule 4 whose conclusion matches with "the engine is getting gas" (labeled with c in figure 5) can be fired. This causes the premises of rule 4 to be placed in the working memory as shown in figure 6.

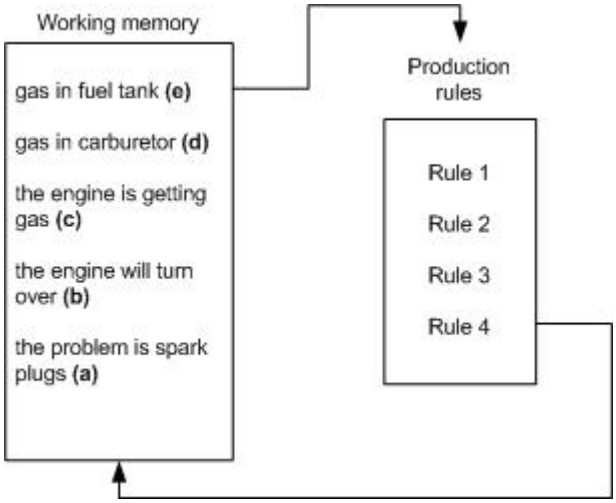

Figure 6: The production system after Rule 4 has fired

At this point, there are three entries (labeled with b, d, and e in figure 6) in working memory that do not match with any rule conditions. In such a situation, the example expert system queries the user about these subgoals. If the user confirms all three of these as true, the expert system will have successfully determined that the car will not start because the spark plugs are bad.

A rule-based expert system can query users through its explanation subsystem shown in figure 1. In general, two questions –why and how- are answered by rule-based expert systems. A "why" query occurs when the program asks the user for information and the user responds with "why", meaning "why did you ask for that information?". The

answer is the current rule that the production system is attempting to fire. The answer to "how did you get that result?" is the sequence of rules that were used to conclude a goal.

In the example, the following dialogue begins with the expert system asking the user about the goals present in working memory.

Expert System: *gas in fuel tank?* (labeled with e in figure 4)
User: *yes*
Expert System: *gas in carburetor?* (labeled with d in figure 4)
User: *yes*
Expert System: *engine will turn over?*
User: *why*

The user wants the expert system to answer the question "why do you want to know whether the engine will turn over?" by entering the query "why". The response of the system may be as follows:

It has been established that:
1. the engine is getting gas,
therefore if
2. the engine will turn over,
then the problem is spark plugs.

Since the user answered yes to the first two questions in the dialogue, the premises of rule 4 (labeled with d and e in figure 6) are satisfied and rule 4 can be fired. This verifies the first premise of rule 1 (labeled with c in figure 6). When the "why" query was made, the system was trying to solve the second premise of rule 1.

If the user asks "how the engine is getting gas?", the system will respond to how it concluded that the engine is getting gas.

This follows from rule 4:
if
gas in fuel tank, and
gas in carburetor
then
engine is getting gas.
gas in fuel tank was given by the user
gas in carburetor was given by the user

In summary, the knowledge-based system answers "why" queries by showing the current rule that it is attempting to fire; it answers "how" queries by giving a trace of the reasoning that led to a goal.

8. Model-based Reasoning

Through years of experience, human experts develop very powerful rules for dealing with commonly encountered situations. These rules take the form of direct associations between observable symptoms and final diagnoses [3].

The MYCIN expert system is an example of model-based reasoning systems. It would propose a diagnosis based on such observable as “headaches”, “nausea”, or “high fever”. Although these parameters can be indicative of an illness, rules that link them directly to a diagnosis do not reflect any deeper, causal understanding of human physiology. MYCIN’s rules indicate the results of an infection, but do not explain its causes [3].

9. Case-based Reasoning

Case-based reasoning uses solutions of previously solved problems as a basis for solving new, but similar, problems. It therefore follows the approach of a human solving a problem from his past experience by analogy.

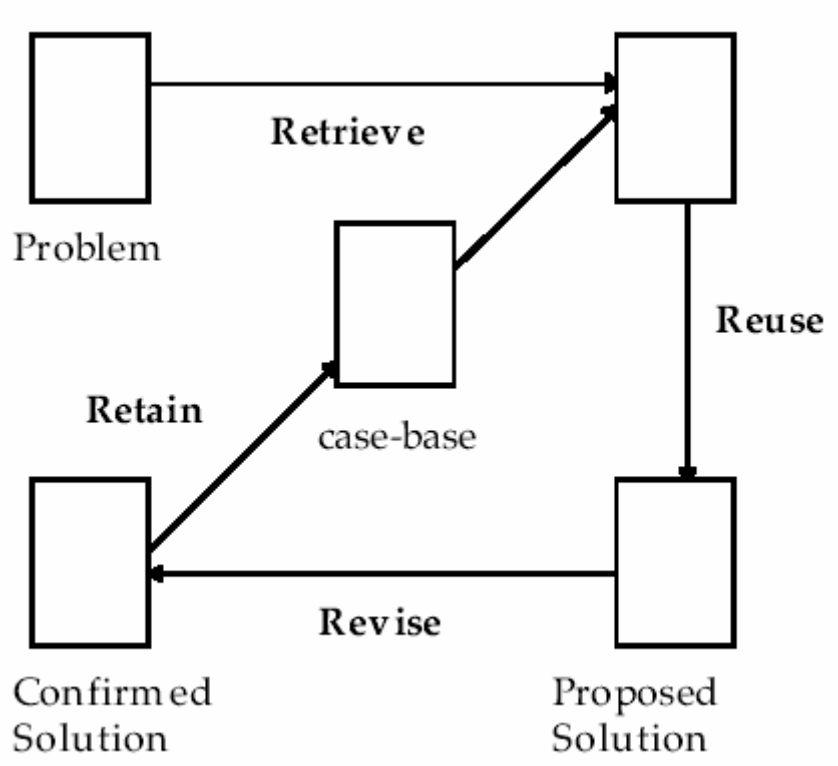

Figure 7: The CBR Cycle

A CBR system normally performs the following four actions cyclically as shown in figure 7:
Retrieve: get the closest match(es) to the problem;
Reuse: use the retrieved cases to solve the problem;
Revise: modify the retrieved solution if necessary;
Retain: save the new solution as a new case.

It is usual that there is human interaction for deciding how to revise and whether to save the new case. So most tools act primarily for the first two actions and the manager of the case-base controls the last two actions.

10. References

[1] Nils J. Nilsson, Artificial Intelligence, Morgan Kaufmann Publishers, 1998.
[2] H. Levent Akın, Principles of AI Course Lecture Notes, Boğaziçi University.
[3] George F. Luger, William A. Stubblefield, Artificial Intelligence, 3rd Edition, Addison-Wesley, 1998.
[4] Efraim Turban, Expert Systems and Applied Artificial Intelligence.