Table 2.1 Comparison of expert systems with conventional systems and human experts

Human experts	Expert systems	Conventional programs
Use knowledge in the form of rules of thumb or heuristics to solve problems in a narrow domain.	Process knowledge expressed in the form of rules and use symbolic reasoning to solve problems in a **narrow domain**.	Process data and use algorithms, a series of well-defined operations, to solve general numerical problems.
In a human brain, knowledge exists in a compiled form.	Provide a **clear separation of knowledge from its processing**.	Do not separate knowledge from the control structure to process this knowledge.
Capable of explaining a line of reasoning and providing the details.	**Trace the rules fired** during a problem-solving session and **explain how** a particular conclusion was reached and **why** specific data was needed.	Do not explain how a particular result was obtained and why input data was needed.
Use inexact reasoning and can deal with incomplete, uncertain and fuzzy information.	Permit **inexact reasoning** and can deal with incomplete, uncertain and fuzzy data.	Work only on problems where data is complete and exact.
Can make mistakes when information is incomplete or fuzzy.	**Can make mistakes** when data is incomplete or fuzzy.	Provide no solution at all, or a wrong one, when data is incomplete or fuzzy.
Enhance the quality of problem solving via years of learning and practical training. This process is slow, inefficient and expensive.	Enhance the quality of problem solving by adding new rules or adjusting old ones in the knowledge base. When new knowledge is acquired, **changes are easy** to accomplish.	Enhance the quality of problem solving by changing the program code, which affects both the knowledge and its processing, making changes difficult.

The characteristics of expert systems discussed above make them different from conventional systems and human experts. A comparison is shown in Table 2.1.

2.6 Forward chaining and backward chaining inference techniques

In a rule-based expert system, the domain knowledge is represented by a set of IF-THEN production rules and data is represented by a set of facts about the current situation. The inference engine compares each rule stored in the

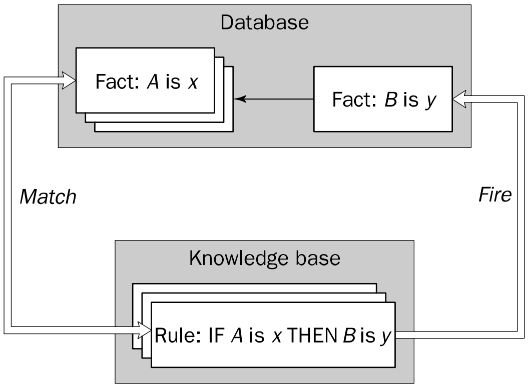

Figure 2.4 The inference engine cycles via a match-fire procedure

knowledge base with facts contained in the database. When the IF (condition) part of the rule matches a fact, the rule is **fired** and its THEN (action) part is executed. The fired rule may change the set of facts by adding a new fact, as shown in Figure 2.4. Letters in the database and the knowledge base are used to represent situations or concepts.

The matching of the rule IF parts to the facts produces **inference chains**. The inference chain indicates how an expert system applies the rules to reach a conclusion. To illustrate chaining inference techniques, consider a simple example.

Suppose the database initially includes facts *A*, *B*, *C*, *D* and *E*, and the knowledge base contains only three rules:

Rule 1:	IF	*Y* is true
	AND	*D* is true
	THEN	*Z* is true
Rule 2:	IF	*X* is true
	AND	*B* is true
	AND	*E* is true
	THEN	*Y* is true
Rule 3:	IF	*A* is true
	THEN	*X* is true

The inference chain shown in Figure 2.5 indicates how the expert system applies the rules to infer fact *Z*. First Rule 3 is fired to deduce new fact *X* from given fact *A*. Then Rule 2 is executed to infer fact *Y* from initially known facts *B* and *E*, and already known fact *X*. And finally, Rule 1 applies initially known fact *D* and just-obtained fact *Y* to arrive at conclusion *Z*.

An expert system can display its inference chain to explain how a particular conclusion was reached; this is an essential part of its explanation facilities.

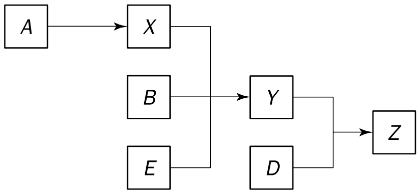

Figure 2.5 An example of an inference chain

The inference engine must decide when the rules have to be fired. There are two principal ways in which rules are executed. One is called **forward chaining** and the other **backward chaining** (Waterman and Hayes-Roth, 1978).

2.6.1 Forward chaining

The example discussed above uses forward chaining. Now consider this technique in more detail. Let us first rewrite our rules in the following form:

Rule 1: $Y \,\&\, D \rightarrow Z$

Rule 2: $X \,\&\, B \,\&\, E \rightarrow Y$

Rule 3: $A \rightarrow X$

Arrows here indicate the IF and THEN parts of the rules. Let us also add two more rules:

Rule 4: $C \rightarrow L$

Rule 5: $L \,\&\, M \rightarrow N$

Figure 2.6 shows how forward chaining works for this simple set of rules.

Forward chaining is the **data-driven** reasoning. The reasoning starts from the known data and proceeds forward with that data. Each time only the topmost rule is executed. When fired, the rule adds a new fact in the database. Any rule can be executed only once. The match-fire cycle stops when no further rules can be fired.

In the first cycle, only two rules, Rule 3: $A \rightarrow X$ and Rule 4: $C \rightarrow L$, match facts in the database. Rule 3: $A \rightarrow X$ is fired first as the topmost one. The IF part of this rule matches fact A in the database, its THEN part is executed and new fact X is added to the database. Then Rule 4: $C \rightarrow L$ is fired and fact L is also placed in the database.

In the second cycle, Rule 2: $X \& B \& E \rightarrow Y$ is fired because facts B, E and X are already in the database, and as a consequence fact Y is inferred and put in the database. This in turn causes Rule 1: $Y \& D \rightarrow Z$ to execute, placing fact Z in the database (cycle 3). Now the match-fire cycles stop because the IF part of Rule 5: $L \& M \rightarrow N$ does not match **all** facts in the database and thus Rule 5 cannot be fired.

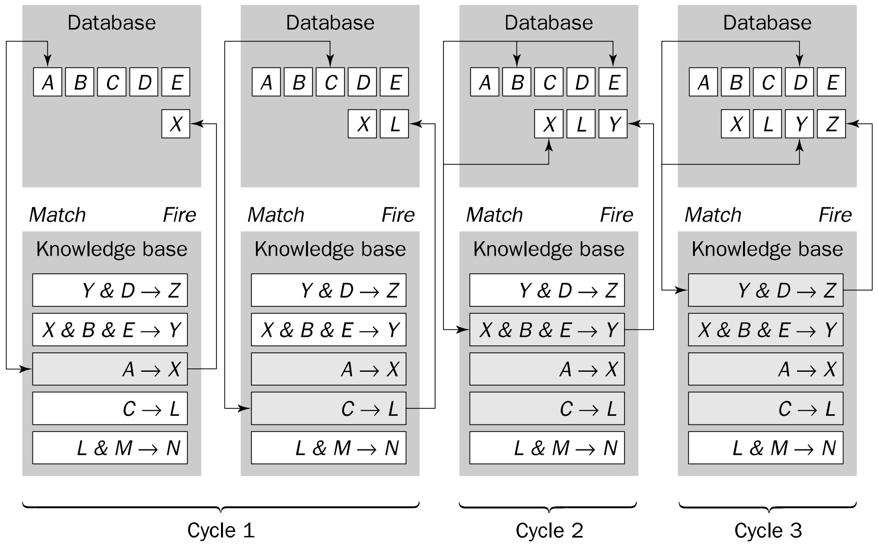

Figure 2.6 Forward chaining

Forward chaining is a technique for gathering information and then inferring from it whatever can be inferred. However, in forward chaining, many rules may be executed that have nothing to do with the established goal. Suppose, in our example, the goal was to determine fact *Z*. We had only five rules in the knowledge base and four of them were fired. But Rule 4: $C \rightarrow L$, which is unrelated to fact *Z*, was also fired among others. A real rule-based expert system can have hundreds of rules, many of which might be fired to derive new facts that are valid, but unfortunately unrelated to the goal. Therefore, if our goal is to infer only one particular fact, the forward chaining inference technique would not be efficient.

In such a situation, backward chaining is more appropriate.

2.6.2 Backward chaining

Backward chaining is the **goal-driven** reasoning. In backward chaining, an expert system has the goal (a hypothetical solution) and the inference engine attempts to find the evidence to prove it. First, the knowledge base is searched to find rules that might have the desired solution. Such rules must have the goal in their THEN (action) parts. If such a rule is found and its IF (condition) part matches data in the database, then the rule is fired and the goal is proved. However, this is rarely the case. Thus the inference engine puts aside the rule it is working with (the rule is said to **stack**) and sets up a new goal, a sub-goal, to prove the IF part of this rule. Then the knowledge base is searched again for rules that can prove the sub-goal. The inference engine repeats the process of stacking the rules until no rules are found in the knowledge base to prove the current sub-goal.

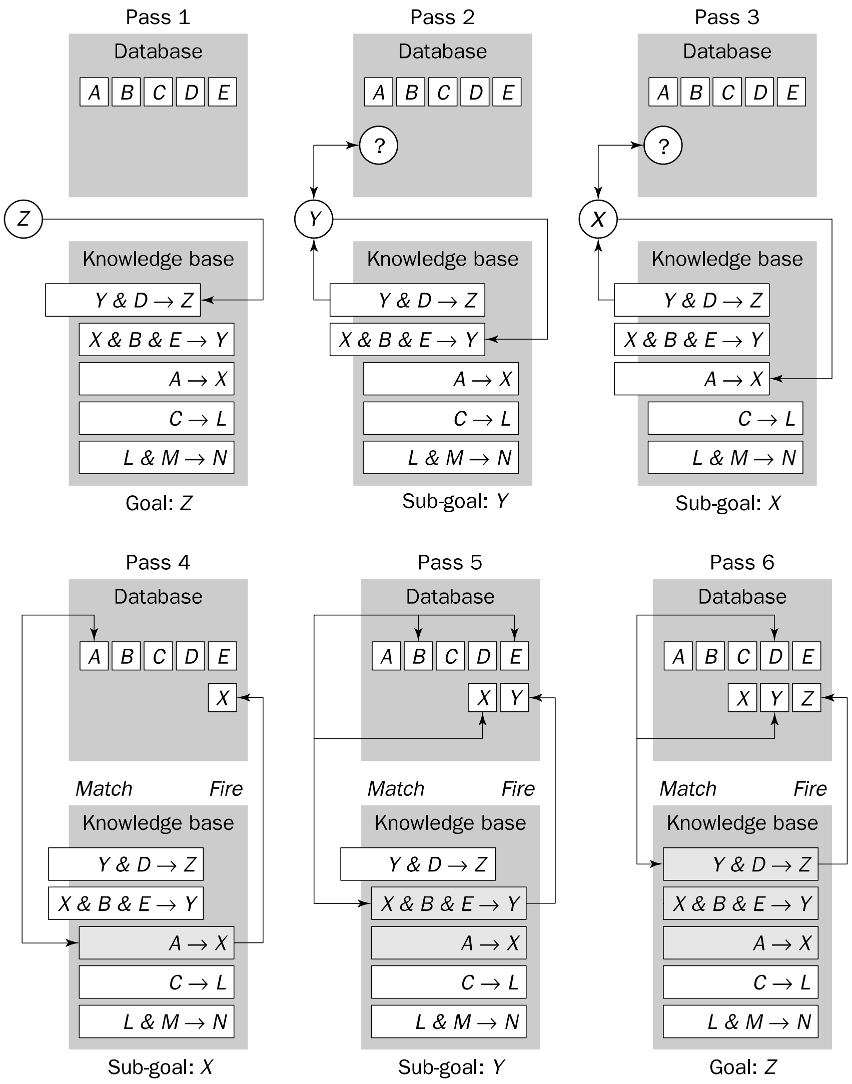

Figure 2.7 Backward chaining

Figure 2.7 shows how backward chaining works, using the rules for the forward chaining example.

In Pass 1, the inference engine attempts to infer fact *Z*. It searches the knowledge base to find the rule that has the goal, in our case fact *Z*, in its THEN part. The inference engine finds and stacks Rule 1: $Y \& D \rightarrow Z$. The IF part of Rule 1 includes facts *Y* and *D*, and thus these facts must be established.

In Pass 2, the inference engine sets up the sub-goal, fact *Y*, and tries to determine it. First it checks the database, but fact *Y* is not there. Then the knowledge base is searched again for the rule with fact *Y* in its THEN part. The inference engine locates and stacks Rule 2: $X \& B \& E \rightarrow Y$. The IF part of Rule 2 consists of facts *X*, *B* and *E*, and these facts also have to be established.

In Pass 3, the inference engine sets up a new sub-goal, fact *X*. It checks the database for fact *X*, and when that fails, searches for the rule that infers *X*. The inference engine finds and stacks Rule 3: $A \rightarrow X$. Now it must determine fact *A*.

In Pass 4, the inference engine finds fact *A* in the database, Rule 3: $A \rightarrow X$ is fired and new fact *X* is inferred.

In Pass 5, the inference engine returns to the sub-goal fact *Y* and once again tries to execute Rule 2: $X \& B \& E \rightarrow Y$. Facts *X*, *B* and *E* are in the database and thus Rule 2 is fired and a new fact, fact *Y*, is added to the database.

In Pass 6, the system returns to Rule 1: $Y \& D \rightarrow Z$ trying to establish the original goal, fact *Z*. The IF part of Rule 1 matches all facts in the database, Rule 1 is executed and thus the original goal is finally established.

Let us now compare Figure 2.6 with Figure 2.7. As you can see, four rules were fired when forward chaining was used, but just three rules when we applied backward chaining. This simple example shows that the backward chaining inference technique is more effective when we need to infer one particular fact, in our case fact *Z*. In forward chaining, the data is known at the beginning of the inference process, and the user is never asked to input additional facts. In backward chaining, the goal is set up and the only data used is the data needed to support the direct line of reasoning, and the user may be asked to input any fact that is not in the database.

How do we choose between forward and backward chaining?

The answer is to study how a domain expert solves a problem. If an expert first needs to gather some information and then tries to infer from it whatever can be inferred, choose the forward chaining inference engine. However, if your expert begins with a hypothetical solution and then attempts to find facts to prove it, choose the backward chaining inference engine.

Forward chaining is a natural way to design expert systems for analysis and interpretation. For example, DENDRAL, an expert system for determining the molecular structure of unknown soil based on its mass spectral data (Feigenbaum *et al.*, 1971), uses forward chaining. Most backward chaining expert systems are used for diagnostic purposes. For instance, MYCIN, a medical expert system for diagnosing infectious blood diseases (Shortliffe, 1976), uses backward chaining.

Can we combine forward and backward chaining?

Many expert system shells use a combination of forward and backward chaining inference techniques, so the knowledge engineer does not have to choose between them. However, the basic inference mechanism is usually backward chaining. Only when a new fact is established is forward chaining employed to maximise the use of the new data.

2.7 MEDIA ADVISOR: a demonstration rule-based expert system

To illustrate some of the ideas discussed above, we next consider a simple rule-based expert system. The Leonardo expert system shell was selected as a tool to build a decision-support system called MEDIA ADVISOR. The system provides advice on selecting a medium for delivering a training program based on the trainee's job. For example, if a trainee is a mechanical technician responsible for maintaining hydraulic systems, an appropriate medium might be a workshop, where the trainee could learn how basic hydraulic components operate, how to troubleshoot hydraulics problems and how to make simple repairs to hydraulic systems. On the other hand, if a trainee is a clerk assessing insurance applications, a training program might include lectures on specific problems of the task, as well as tutorials where the trainee could evaluate real applications. For complex tasks, where trainees are likely to make mistakes, a training program should also include feedback on the trainee's performance.

Knowledge base

```
/* MEDIA ADVISOR: a demonstration rule-based expert system

Rule: 1
if      the environment is papers
or      the environment is manuals
or      the environment is documents
or      the environment is textbooks
then    the stimulus_situation is verbal

Rule: 2
if      the environment is pictures
or      the environment is illustrations
or      the environment is photographs
or      the environment is diagrams
then    the stimulus_situation is visual

Rule: 3
if      the environment is machines
or      the environment is buildings
or      the environment is tools
then    the stimulus_situation is 'physical object'

Rule: 4
if      the environment is numbers
or      the environment is formulas
or      the environment is 'computer programs'
then    the stimulus_situation is symbolic
```