EXPERT SYSTEMS FOR EXPERTS

Kamran Parsaye
IntelligenceWare, Inc.
Los Angeles, California

Mark Chignell
Department of Industrial and Systems Engineering
University of Southern California
Los Angeles, California

Reprinted by permission of the publisher from: Parsaye, Kamran. *Expert Systems for Experts*. New York: John Wiley and Sons, Chap. 2, p. 48-57.

WILEY

John Wiley & Sons, Inc.
New York ▪ Chichester ▪ Brisbane

2.3.2.2 *Metarules* Although rules represent a fairly simple knowledge structure, they can be used in a number of different ways. For instance, some rules can be designed to control the behavior of other rules. Imagine the case of a small company that begins to expand. Initially all the employees are directly involved in basic activities such as production, development, and marketing. Later, as the company becomes more complex and more employees are hired, some of the employees begin to manage the activity of other employees.

The same sort of thing can happen in an expert system. When there are only a few rules in the knowledge base, each rule represents part of the knowledge domain. As an increasing number of rules are added, an overhead develops for organizing the activities of the rules. Rules, known as *metarules,* are developed that manage the activities of other rules. An example of a metarule in a financial advisor might be:

If
    the age of the client is greater than 65
and
    there are rules that mention blue chip risk in their premise
and
    there are rules that mention speculative risk in their premise
Then
    use the former set of rules before using any of the latter;

Building this type of *metaknowledge* into the knowledge base can be difficult. One problem with metarules is that there will always be exceptions. In the preceding example, for instance, there may be a class of senior citizens who already have a secure income and who would be interested in speculative risks.

2.3.3 Frames: Packaged Structures

Facts and rules are important knowledge structures, but we also need a way of packaging knowledge that makes it easily accessible. Packages provide modularity, hierarchical organization, and compactness of expression.

Hierarchical organization enhances modularity by allowing us to describe or refer to a class of concepts using a single high-level representative of that class. Hierarchies also assist the system in knowing where to look for information. For instance, finding information about automobiles begins by looking in the automobile section of the hierarchy. Compact expression is achieved by having to define something only once. It can then be shared by lower-level instances of the same concept.

Packaged structures can assist the expert in reasoning in a variety of ways. They make it easier to organize and retrieve knowledge. In analyzing hazards, for instance, the expert knows what information, such as voltage, is important

and what sorts of situations may lead to an electrical hazard. Packaged structures can also guide the expert in carrying out the task by specifying the important information that should be known about an object. All hazards, for instance, will have causes, effects, and possible strategies for controlling or removing them, and for each hazard, this type of information should be packaged with other information about that hazard.

Structures seem to be a natural way of organizing knowledge. There is a considerable amount of evidence that human memory contains knowledge structures that aid in recall of information. An early study of human long-term memory (Bartlett, 1932) demonstrated the existence of patterns derived from previous experience that are used in interpreting new experiences. Bartlett used the term "schema" to refer to these patterns. Packaged knowledge structures in machines perform similar functions in organizing and interpreting knowledge.

Consider the concept of an automobile. Most people already know a great deal about what an automobile should be like and have concepts such as:

Automobiles are a type of vehicle

Automobiles carry passengers

An automobile requires a driver

An automobile uses gas for energy

These statements represent reasonable expectations about what properties an automobile should have. However, listing general statements about automobiles in this fashion does not show how they are related to each other. Instead, we need to package this information into a more usable form. Thus we might create a standardized form of representation that captures the critical elements of the automobile structure in a way that allows the structure to be related to other structures. We might, for instance, standardize our general notions about automobiles in the form shown in Figure 2.12.

In representing knowledge about a domain, a knowledge base needs to store knowledge in a form that can be used for efficient storage, retrieval, and reasoning. One knowledge structure that fulfills these purposes and which is frequently used in expert systems is the *frame*. Frames are a way of packaging knowledge within a well defined structure.

The basic idea of a frame was outlined by Minsky (1975), who in introducing the notion of frames, wrote:

> . . . the ingredients of most theories both in Artificial Intelligence and in Psychology have been on the whole too minute, local, and unstructured to account . . . for the effectiveness of common-sense thought. The "chunks" of

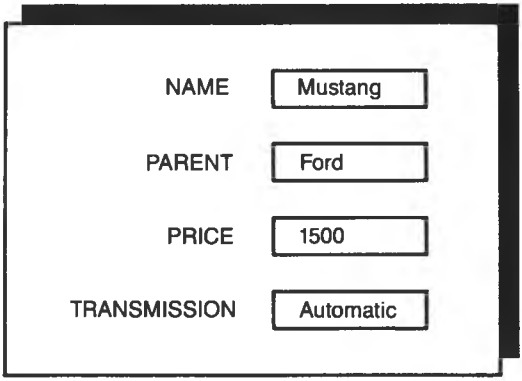

Figure 2.12 A "packaged" version of basic knowledge about an automobile.

reasoning, language, memory, and perception ought to be larger and more structured; their factual and procedural contents must be more intimately connected in order to explain the apparent power and speed of mental activities.

Frames were seen as a way round this problem. A frame is a specialized structure that represents a stereotypical situation. Additional power is added to the frame concept by allowing information to be attached to each frame. These attachments can include instructions about how to use the frame, what should happen next, and what to do if the expectations are not confirmed.

Each frame has a higher-level frame (parent frame) to which it belongs. For instance, the parent frame for Ford may be Automobile. The characteristics of each frame are captured in its *slots* or *attributes*. A frame may contain a number of *slots* that can be filled with specific instances or data.

Consider automobiles and their role in transport. The following information describes some relevant knowledge about automobiles:

An automobile is a land vehicle

A Ford is an automobile

A land vehicle is a vehicle

A vehicle is a form of transport

An automobile is powered by an engine

An automobile can usually carry from 2 to 6 passengers

Information about automobiles, and about vehicles in general, can be linked into a hierarchy, as is shown in Figure 2.13. The description of our knowledge about automobiles should also organize the knowledge in a way that includes the information in each statement listed above.

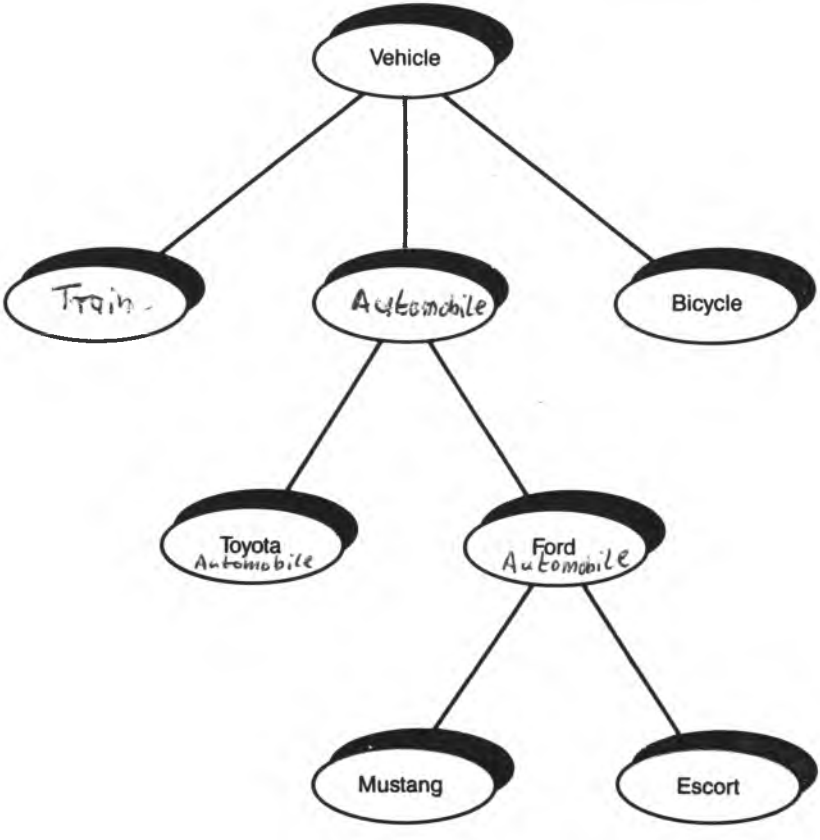

Figure 2.13 A hierarchy representing knowledge about automobiles.

The knowledge structures shown in Figure 2.13 suggest that vehicles share the function of transportation and that land vehicles share the function of transportation over land. The frames become more specific as we move down the hierarchy. Vehicles are manufactured in all shapes and sizes, however, when we get down to the more specific level of automobiles we have a fairly strong set of expectations about an automobile's structure and function.

Automobiles also have general attributes such as the engine size, the capacities of the passenger and luggage compartments, the mileage on the speedometer, the size of the tires, and so on. These shared attributes can be expressed as slots in the general automobile frame. Specific automobiles will have additional attributes that are not shared with other automobiles. While most automobiles have four wheels, relatively few are convertible, have sun roofs or heavily tinted windows.

Much of the power of frames as knowledge representation tools is derived from their ability to handle default reasoning. In the absence of external infor-

mation, a slot in a frame can be filled with a default value that will be assumed until new information is obtained. Thus we might assume that a power plant burns coal, or that a teacher has a college degree until told otherwise, or that all automobiles have four wheels.

If we establish the syntactic structure for frames:

Frame Name
Parent Frame
slot1 filler1
slot2 filler2
. . .
slotN fillerN

then we can develop an equivalent pictorial representation. The frame representations of vehicles and automobiles is illustrated in Figure 2.14.

An important aspect of this frame hierarchy is that general attributes described by slots in higher-level frames are shared by all automobiles and specific attributes are stored locally in slots in lower level frames (with each particular instance of an automobile). The relationship between Figures 2.13 and 2.14 is very close. Each parent slot in a frame in Figure 2.14 is equivalent to the link between corresponding nodes in Figure 2.13.

Using the frame hierarchy in Figure 2.14, we can draw conclusions about a particular Ford Escort based on the fact that it is a Ford and that it is an automobile. Knowing that it is an automobile, we can conclude that the 1983 hatchback model has four wheels and an engine unless told otherwise.

We can build a hierarchy of frames where the topmost nodes represent general concepts and the lower nodes represent more specific instances of those concepts. This is done by connecting frames in a series of *parent–child* relationships. Thus the parent frame for Ford is Automobile.

Each frame thus represents a concept as a collection of attributes that are referred to as *slots.* Slots can be filled with *values;* the value of the Engine slot for the Ford Escort frame is 4-Cylinder.

If we want to collect information from a frame, and we don't find that information in the frame itself, we can try and obtain the information from its parent and so on using the process of *inheritance.* For instance, suppose that we tried to find out how many wheels a Ford Escort has using the frame hierarchy shown in Figure 2.15. Since this information is not stored in the Ford Escort frame, we try its parent, but the information is not present in the Ford frame, either. We then move to Automobile, the parent of Ford, and it is here that we get the answer, i.e., that there are four wheels. We can paraphrase the kind of reasoning involved in this example of inheritance as follows:

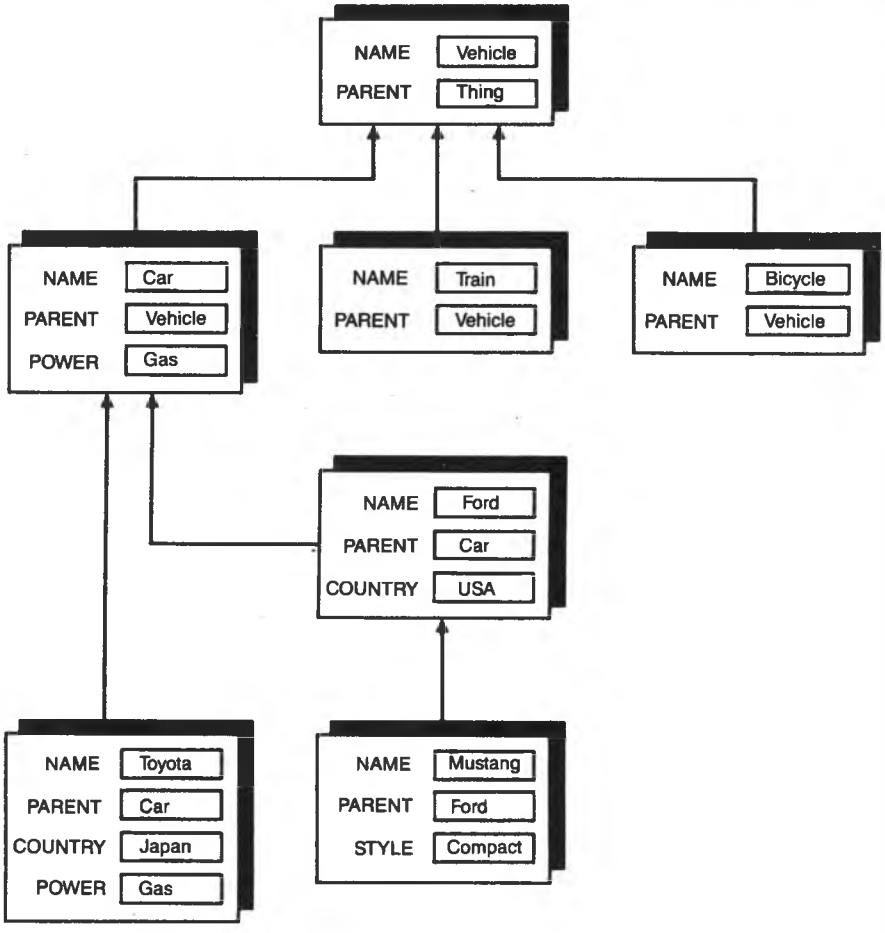

Figure 2.14 A frame representation of the automobile hierarchy.

Since I know that a Ford Escort is a Ford
and I know that a Ford is an automobile
and I know that an automobile has 4 wheels
I can assume that a Ford Escort has 4 wheels

Inheritance saves us the trouble of explicitly going through this type of reasoning process each time we want to retrieve information from a frame hierarchy. Inheritance is a natural consequence of using knowledge structures organized in hierarchies. One uses a hierarchy by placing information at the highest level where it can reasonably be expressed. It can then be shared by all the concepts that are children or descendants of the concept that contains the information.

Inheritance is the process of making the information stored in a high-level concept available to lower-level instances of that concept. Inheritance works well when knowledge is organized into a network or hierarchy. Inheritance is both a structuring principle and a set of processes that search for information that is not immediately available.

The process of inheritance that answers the question "how many wheels does a Ford Escort have?" does so by looking up the information in a set of frames. We begin by looking for the information in the Ford Escort frame; we then look for it in the parent (Ford) and finally the parent's parent (automobile), where, in fact, we find the information. This inheritance process is illustrated in Figure 2.15.

Inheritance should not immediately be confused with organized facts. The following facts, for instance, describe a portion of a hierarchy, but they do not provide a mechanism for inheriting the number of wheels that a Ford has on the basis of the number of wheels that an automobile has:

A Ford Escort is a Ford

A Ford is an Automobile

An Automobile has four wheels

An automobile has an engine

If we use facts to represent a hierarchical structure we must then use rules for inheriting information. For instance, we could find out the number of wheels that a Ford has by applying the following rules for inheriting that information:

```
If
        'Object' is a Ford Escort
Then
        'Object' is a Ford;
If
        'Object' is a Ford
Then
        'Object' is an automobile;
If
        'Object' is an automobile
Then
        'Object' has four wheels;
```

However, it is more convenient to use frames, since we would have to create a similar rule for each step in the inheritance path, and we wouldn't have the packaging of information that frames provide.

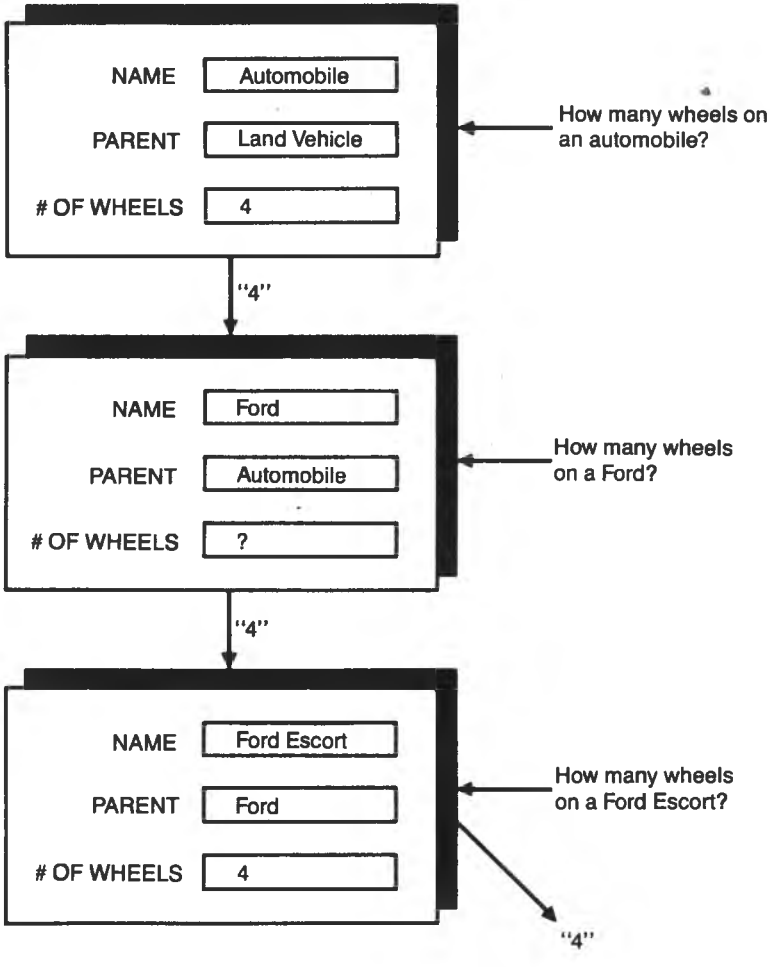

Figure 2.15 An example of inheritance in frames.

Rules and frames should not be seen as competing ways of representing knowledge. Frames are a good way of packaging knowledge and handling the storage and retrieval of that knowledge. Rules, on the other hand, work best at making deductions. The functionality of frames can certainly be implemented in rules, but this creates a whole new set of extra rules, and the resulting rule system will be harder to work with. Further, implementing frames in rules results in computational performance degradation and slow response time.

We have now introduced three important features that frames provide:

Hierarchical representation.
Attribute packaging.
Inheritance.

There is one more feature still to be introduced. What happens if the system needs some information but that information is not available, even after inheritance has been tried? One solution is to allow the system to somehow determine the information, by asking someone, proving a rule, etc. Thus we might have a rule called "Go ask for it" that is invoked when the original search for the information in the frame hierarchy fails. In a frame hierarchy, this type of behavior is achieved by attaching the "Go ask for it" rule to the appropriate slot in the frame. A rule such as "Go ask for it" is generally referred to as an *attached* procedure, or an *attached predicate.*

Consider the frame for an automobile which has been modified to include an attached predicate on the fuel consumption slot, as shown in Figure 2.16.

In this case we are trying to obtain information from the frame, so the attached procedure is referred to as an *if-needed* procedure. Now, if the system needs to find the fuel consumption of a Ford Escort, it looks in the Ford Escort frame, draws a blank, and then works its way up the hierarchy using inheritance. When it gets to the fuel consumption slot in the automobile frame, it finds the if-needed rule and then uses "Go ask for it" to get the information it needs.

If-needed procedures are used when information is needed from a slot but the slot is empty. Several other types of attached procedure have been suggested for frame systems. An *if-added* procedure, for instance, is invoked when the system attempts to add something to a slot.

It should be clear from this introduction that frames provide a versatile way of organizing knowledge. In practice, the majority of expert system tasks can

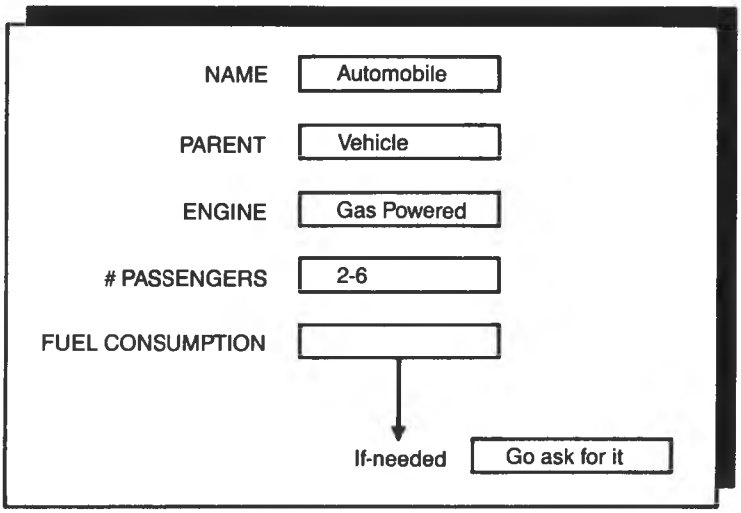

Figure 2.16 A frame containing an attached procedure.

be efficiently implemented using rules or frames. However, since frames are a good way to manage knowledge and rules are well suited to reasoning, it seems natural to combine rules and frames when building expert systems.

■ 2.4 REASONING WITH KNOWLEDGE

Expert systems work with knowledge structures and rules of knowledge, but how? An expert system typically includes:

Facts, which are elementary pieces of knowledge.

Frames, used to organize knowledge.

Rules, which relate facts and frames.

We collectively refer to these facts, rules, and frames as the knowledge the expert system possesses. To arrive at conclusions, the expert systems needs to relate pieces of knowledge by performing *inference* or deduction. The part of an expert system that performs inference is called an *inference engine.*

The task of the inference engine is to take the knowledge in the knowledge base and carry out a set of actions that will utilize the knowledge in finding a solution to the problem. The same knowledge may be used for performing different types of task.

Human experts are able to use similar pieces of knowledge in more than one way. For instance, knowledge of medicine can be used for diagnosing a patient, prescribing treatments, or suggesting preventive regimens.

Consider how inference works for a safety expert who is dealing with garage fires and related hazards. The expert has a core of knowledge to be used for a number of different purposes. The expert may use this knowledge to diagnose why a fire took place, to predict the existence of a hazard, or to recommend how hazards may be eliminated.

Imagine that a garage fire has just occurred. In searching for a cause of the fire, the expert discovers that a set of fine cutlery had been left in the garage prior to the fire and now suspects a cause for the fire. The expert knows that:

Silver is a precious metal

Fine cutlery contains silver

Radiator coolant contains Ethylene-Glycol

Asbestos particles are potent carcinogens

Radiator coolant is used in servicing automobiles

Automobiles are housed in garages

etc.