Knowledge Representation and Software Selection for Expert Systems Design

ARDESHIR FAGHRI AND MICHAEL J. DEMETSKY

A variety of techniques and methods for representing knowledge in the knowledge base of expert systems have been used. The authors examine the significance of the means of representing the knowledge base in the development of expert systems, with special reference to transportation engineering. The development, highlights, and shortcomings of each representation technique are discussed, and appropriate transportation engineering examples are given. Also presented are the results of an investigation of expert system tools and how they relate to different representation techniques.

The heart of an expert system is its knowledge, which is structured to support decision making. When scientists in artificial intelligence (AI) use the term "knowledge," they mean the information a computer needs before it can function intelligently (*1*); this information takes the form of facts and rules. Facts are truths in some relevant world—things we want to represent. Representations of facts are the things we will actively be able to manipulate. For example,

Fact: Responses to a brake light from a leading vehicle require 0.4 sec to more than 1.0 sec for some drivers (*2*).

Example: All physical motor capabilities deteriorate with age.

Rules are formal representations of recommendations, directives, and strategies; they may be expressed as conditional (if-then) statements. For example,

Rule: If forced flow and low speeds exist on a segment of highway, a level of service F is achieved.

Example: If the degree of congestion or vehicle delay, or both, caused by daytime lane closures is severe, nighttime construction and maintenance should be considered.

Facts and rules in an expert system are not always true or false; sometimes there is a degree of uncertainty about the truth of a fact or the validity of a rule. When this doubt is made explicit, it is called a certainty factor (*1*).

Fact: Rail–highway crossings near major employment centers experience more accidents with certainty 0.7.

Virginia Transportation Research Council, Box 3817, University Station, Charlottesville, Va. 22903.

Rule: If the average speed increases by 10 mph with certainty 1.0, the number of accidents will increase by 10 percent with certainty 0.6.

The organization of knowledge in an expert system separates the knowledge about the problem domain from the system's other knowledge, such as general knowledge about how to solve problems or knowledge about how to interact with the user. The collection of domain knowledge is called the knowledge base; the general problem-solving knowledge is called the inference engine (*1*).

Different techniques and methods for representing knowledge in the knowledge base of expert systems have been used. The authors examine the significance of the means of representing the knowledge base in the development of expert systems, with special reference to transportation engineering. The development, highlights, and shortcomings of each representation technique are discussed, and appropriate transportation engineering examples are given. Also presented are the results of a thorough investigation of expert system tools and how they relate to different representation techniques.

KNOWLEDGE REPRESENTATION TECHNIQUES

In expert systems, complex problem solving requires both a large amount of knowledge and a mechanism for manipulating that knowledge to create solutions to new problems. A number of methods for representing knowledge (facts) have been used in expert systems. In this paper, the common knowledge representation techniques are discussed: predicate logic, other logics, structured representation, rules, and object-attribute-value triplets.

Predicate Logic

Logic is critically concerned with the validity of arguments, that is, methods of determining whether given conclusions can be validly drawn from given facts. Logic is relevant to programming because a program is really a set of quasi-logical statements that are processed in some way to generate a conclusion (*3*). In logic a "true argument" has a

precise, clearly defined meaning: an argument is considered true if and only if all of its assumptions are true; then its conclusions are true also.

To decide on the acceptability of a particular argument, it is necessary to make some test. In logic the method of doing this is to compare the text of interest with abstracted patterns of argument to seek a match. Such patterns are termed "forms" and are made up of abstracted sequences of facts and rules that have been proved valid in a mathematical (or "formal") way (*3*).

The capability of logic to generate (or infer) new information from old is of particular interest given the tendency to view programming as the controlled generation of inferences. Moreover, with many years of development behind it, logic also provides a well-defined and well-understood formalism for representing facts and the rules for manipulating them.

Before discussion of the concepts and applications of predicate logic, using propositional logic as a way of representing the sort of world knowledge an expert system might need is explored. Propositional logic is appealing because it is simple to deal with and there is a decision procedure for it. Real-world facts can easily be represented as logical expressions (or logical propositions) written as well-formed formulas (wff's) in propositional logic, such as the following (*4*):

It is raining.
RAINING
It is sunny.
SUNNY

These propositions could be used, for example, to deduce that it is not sunny if it is raining:

If it is raining then it is not sunny.
RAINING –SUNNY

But it is easy to observe the limitations of propositional logic. The obvious fact stated in "an automobile is a vehicle" could be written "Autovehicle." But "a truck is a vehicle" would have to be written "Truckvehicle."

This would be a totally different assertion, and no conclusions could be drawn about similarities between "auto" and "truck." It would be much better to represent these facts as "Vehicle (auto)" and "Vehicle (truck)," because the structure of the representation would reflect the structure of the knowledge itself. It is even more difficult if we try to represent "all vehicles are unsafe," because quantification would be needed unless separate statements were written about the safety of every known vehicle.

So it appears that predicate logic must be the way to represent knowledge, because it permits representations of things that cannot reasonably be represented with propositional logic. In predicate logic, real-world facts can be represented as statements written as wff's. But a major motivation for choosing to use logic was that if logical statements were used as a way of representing knowledge, then there would be a good way to reason with that knowledge. Determining the validity of a proposition in propositional logic is straightforward, though computationally it may be difficult.

Predicate logic provides a way of deducing new statements from old ones. Unfortunately, however, unlike propositional logic, it does not have an associated decision procedure (*4*). There are procedures that will lead to a proof of a proposed theorem (if indeed it is a theorem), but they will not necessarily halt if the proposed statement is not a theorem. One such simple procedure is to use the rules of inference to generate theorems from axioms in some orderly fashion, testing each to see if it is the one for which a proof is sought. This method, however, is not very efficient, and investigation continues to find better ones. So, despite the theoretical undecidability of predicate logic, it can still serve as a useful way of representing and manipulating some of the kinds of knowledge that an expert system might need.

Knowledge representation by using predicate logic is demonstrated by the examples shown below. Consider the following statements:

1. Lee Highway was congested.
2. Lee Highway is an Interstate.
3. All Interstates are highways.
4. Washington Metro is a heavy rail train.
5. All people either like downtown New York or hate it.
6. People only try to ignore freeways that are congested.

The facts described by these sentences can be represented as a set of wff's in predicate logic. But some notation must be defined first:

$\wedge$ = AND
$\vee$ = OR
$-$ = negation
$\forall$ = for every
$\exists$ = there exists

By using this notation, the predicate logic version of the six statements may be presented as follows:

1. Lee Highway was congested.
   Congested (Lee Highway)

This representation captures the critical fact that Lee Highway is congested. It fails to capture some of the information in the English sentence, namely, the notion of past tense. Whether this omission is acceptable depends on how the knowledge is to be used.

2. Lee Highway is an Interstate.
   Interstate (Lee Highway)
3. All Interstates are highways.
   $\forall$ xInterstate(x) $\rightarrow$ Highway(x)
4. Washington Metro is a heavy rail train.
   Heavy rail train (Washington Metro)

Here we ignore the fact that proper names are often not references to unique items, because many things share the same name. Sometimes deciding which of several things is being referred to in a particular statement may require a fair amount of knowledge and reasoning.

5. All people either like downtown New York or hate it.

$\forall$ xPerson(x) $\rightarrow$ like(x, New York) $\vee$ hate(x, New York)

In English, "or" sometimes means the logical inclusive "or" and sometimes means the logical exclusive "or." Here the inclusive "or" is used. Some may argue that this English sentence is really stating an exclusive "or." Its expression would be:

$\forall$ xPerson(x) $\rightarrow$ [like(x, New York) $\vee$ hate(x, New York)]

$\wedge \sim$ [like(x, New York) $\wedge$ hate(x, New York)]

6. People only try to ignore freeways that are congested.

$\forall$ x $\forall$ y person(x) $\wedge$ freeway(y) $\wedge$ tryignore(x, y)

This sentence, too, is ambiguous. Does it mean that the only freeways that people try to ignore are those that are congested (the interpretation used here)? or Does it mean that the only thing people try to do is to ignore congested freeways?

From these statements, three important issues must be addressed when converting English sentences to logical statements and then using those statements to deduce new ones:

1. Many English sentences are ambiguous. Choosing the correct interpretation may be difficult.
2. There is often a choice of ways of representing the knowledge. Simple representations are desirable, but they may preclude certain kinds of reasoning. The useful representation for a particular set of sentences depends on the use to which the knowledge contained in the sentences will be put.
3. Even in very simple situations, a set of sentences is unlikely to contain all the information necessary to reason about the topic at hand.

Although predicate logic is a useful way of representing knowledge for many expert system domains, some kinds of information may not be easily represented by this method. Discussed in the next section are other useful methods of knowledge representation.

Other Logics

The techniques of predicate logic are useful for solving problems in many different domains. But unfortunately in many other interesting domains, predicate logic does not provide a good way of representing and manipulating important information. Such domains are mostly uncertain and fuzzy. The methods discussed in this section for these problems with computer programs are monotonic logic and statistical and probabilistic reasoning.

Monotonic Logic

Monotonic logic allows statements to be deleted from as well as added to the data base. Among other things, it allows belief in one statement to depend on lack of belief in some other one. Rarely does a system contain all the information that would be useful. But often when such information is lacking, some sensible guesses can be made as long as there is no contradictory evidence. The construction of these guesses is known as "default reasoning."

For example, suppose the chief traffic engineer of a large metropolitan city decides to add one lane to a road in the city's street network that has been determined to have severe congestion during peak hours (after all economic and social issues have been resolved). Would adding a lane relieve the congestion? The engineer can approach the problem fairly well if he uses a general rule: Because the peak-hour volume exceeds the capacity of the facility in question, adding an extra lane would relieve the congestion unless there is evidence to the contrary.

This sort of default reasoning is nonmonotonic (i.e., the addition of one piece of information may force the deletion of another), because statements so derived depend on lack of belief in certain other statements. This means that if one of those previously lacking statements is added to the system, the statement generated by default reasoning will have to be deleted. Thus, in our example, if the engineer fails to realize that the fundamental differences between the normative system optimization (SO) flow pattern and the descriptive user equilibrium (UE) flow pattern on the network may lead to Braess's paradox (*5*) (the addition of the lane may actually increase the travel time of the vehicles), he should delete his previous belief that this particular strategy will work. Of course, he must also delete any other beliefs that are based on the belief that has just been discarded. This kind of default reasoning is referred to as the "most probable choice" (*4*).

In general, nonmonotonic reasoning systems may be necessary because of (a) incomplete information, which requires default reasoning; (b) changing knowledge that must be described by a changing data base; or (c) a complete solution to problems, which may require assumptions about partial solutions.

Statistical and Probabilistic Reasoning

In representing the knowledge in expert systems, it is assumed that either a fact is known to be true, or it is known to not be true, or nothing at all is known about it. Still to be considered is the possibility of facts that may be "probably true." There are three kinds of situation in which probabilistic reasoning may be employed:

1. The relevant world is really random, for example, the motion of electrons in an atom or the distribution of speeds on a certain highway.
2. The relevant world is not random given enough data, but a system will not always have access to that many

data, for example, the likelihood of success of a traffic control strategy to combat congestion.

3. The world appears to be random because it has not been described at the proper level.

Probabilistic reasoning in the first two cases is utterly appropriate. The mathematical theory of probability provides a way of describing and manipulating uncertain knowledge. Sometimes very simple techniques of probability can be used effectively in expert systems.

One of the most useful results of probability theory is Bayes's theorem, which provides a way of computing the probability of a particular event given some set of observations. The theorem states:

$$P(H_i \mid E) = \frac{P(E \mid H_i) * P(H_i)}{\sum_{n=1}^{k} P(E \mid H_n) * P(H_n)}$$

where

$P(H_i \mid E)$ = probability that hypothesis i is true given evidence E,
$P(E \mid H_i)$ = probability that evidence E will be observed given that hypothesis i is true,
$P(H_i)$ = a priori probability that hypothesis i is true in the absence of any specific evidence, and
k = number of possible hypotheses.

Bayes's theorem can be modified to handle a variety of more complicated situations. For example, a single body of evidence E might not be collected all at once. Rather, a series of smaller observations might be made over time. Other results in probability theory can also be applied to these kinds of problems.

Structured Representation

A good system of representing complex structured knowledge in a particular domain for use in expert systems should have the following four properties (*4*):

1. Representation adequacy—the ability to represent all of the kinds of knowledge that are needed in that domain.
2. Inferential adequacy—the ability to manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from old.
3. Inferential efficiency—the ability to incorporate into the knowledge structure additional information that can be used to point the inference mechanisms in the most promising directions.
4. Acquisitional efficiency—the ability to acquire new information easily. The simplest case involves direct insertion of new knowledge into the data base. Ideally, the program itself would be able to control knowledge acquisition.

The representation techniques discussed previously are useful for representing simple facts, but they cannot always have the desired properties of a representation technique. Several techniques for acquiring these properties have been developed. These techniques are referred to as "declarative methods" (*4*). In declarative knowledge representation, most of the facts are presented as a static collection of knowledge accompanied by a small set of general procedures for manipulating them. In this section, three declarative mechanisms for representing knowledge are presented: semantic nets, frames, and scripts.

Semantic Nets

The term "semantic net" is used to describe a knowledge representation method that is based on a network structure. Semantic nets were originally developed for use as psychological models of human memory but are now a standard method of representation for artificial intelligence and expert systems. A semantic net consists of points (nodes) connected by links (arcs) describing the relations between the nodes. The nodes in a semantic net represent objects, concepts, or events. Arcs can be defined in different ways, depending on the kind of knowledge being represented.

Isa arcs are most often used to establish a property inheritance hierarchy; that is, instances of one class have all properties of more general classes of which they are members. Has-part arcs identify nodes that are properties of other nodes. Figure 1 shows both isa and has-part arcs in a simple net for the concept of a public transit mode.

The isa relation, like the has-part relation, establishes an inheritance hierarchy for properties in the net (*1*), so that items lower in the net inherit properties from items higher in the net. This saves space, because information about similar nodes does not have to be repeated at each node and can be stored in one central location. For example, in the public transit–mode semantic net the common parts of each node, such as passenger seats and engine, are stored once at the node level instead of repeatedly at lower levels like a bus or a particular bus system. The net can be searched, by using knowledge about the meaning of the relations in the arcs, to establish facts like "Washington Metro has passenger seats." Semantic nets are a useful way to represent knowledge and to simplify problem solving in domains that use well-established taxonomies (*1*).

Frames

Frames provide another method of representing facts and relationships. A frame is a description of an object that contains slots for all the information associated with the object. Values may be stored in slots. Each slot can have any number of procedures attached to it. Three useful kinds of procedure often attached to slots are

1. If-added—executes when new information is placed in the slot,

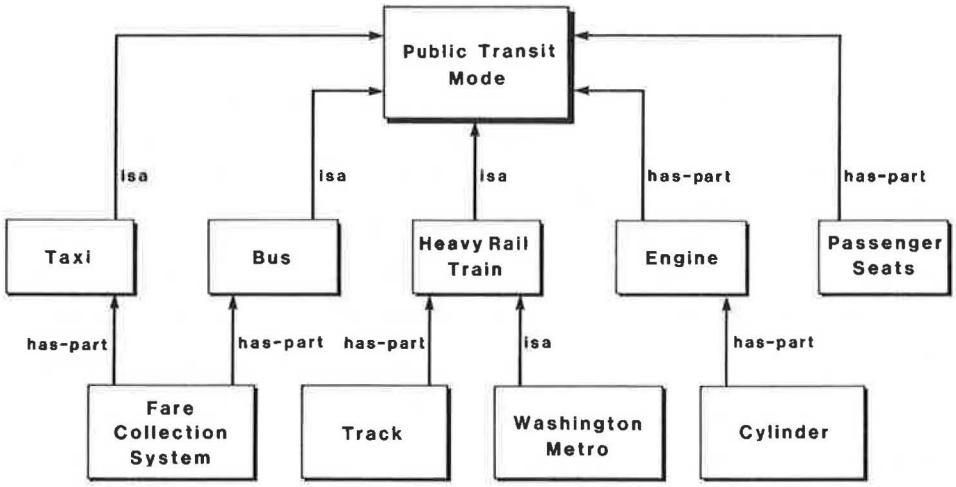

FIGURE 1 Semantic network, showing isa and has-part arcs.

2. If-removed—executes when information is deleted from the slot, and
3. If-needed—executes when information is needed from the slot but the slot is empty.

These attached procedures can monitor the assignment of information to the node, thereby ensuring that appropriate action is taken when values change.

A frame is organized much like a semantic net. It is a network of nodes and relations organized in a hierarchy in which the higher nodes represent general concepts and the lower nodes represent properties of those concepts. Frame systems are useful for problem domains in which expectations about the form and content of the data play an important role in problem solving, such as interpreting visual scenes or understanding speeches. Figure 2 shows an example of a frame network for an expert system.

Scripts

A script is a structure in which a stereotyped sequence of events in a particular context is described. A script consists of a set of slots. Associated with each set of slots may be some information about what kind of values it may contain, as well as a default value to be used if no other information is available. So far this definition seems similar to that for frames, but scripts have other important components, a few of which are

1. Entry conditions—conditions that, in general, must be satisfied before the events described in the script can occur;
2. Results—conditions that, in general, will be true after the events described in the script have occurred;
3. Props—slots that represent objects that are involved in the events described in the script (the presence of these objects can be inferred even if they are not mentioned explicitly);
4. Roles—slots that represent people who are involved in the events described in the script (the presence of these people, too, can be inferred even if they are not mentioned explicitly—if specific individuals are mentioned, they can be inserted into the appropriate slots); and
5. Track—the specific variation on a more general pattern that is represented by the particular script (different tracks of the same script will snare many but not all components).

Although scripts are less general than are frames, and so are not suitable for representing all kinds of knowledge, they can be very effective for representing the specific kinds of knowledge for which they are designed.

Rules

In expert systems the term "rule" refers to the most popular type of knowledge representation technique, the rule-based representation. Rules provide a formal way of representing recommendations, directives, and strategies; they are often appropriate when the domain knowledge results from empirical associations developed through years of problem-solving experience. Rules are generally expressed as conditional (if-then) statements. Rules might exist in an expert system for determining whether to rehabilitate or replace highway bridges:

1. If a bridge has a sufficiency rating between 50 and 80, then it should be rehabilitated.
2. If a bridge is scheduled to be replaced within 6 yr, then only routine maintenance will be necessary until it is replaced.
3. If any one component of a bridge (namely, the substructure, superstructure, or deck) has a condition rating greater than 5 and the bridge is less than 20 years old, than only routine maintenance will be required on that component.

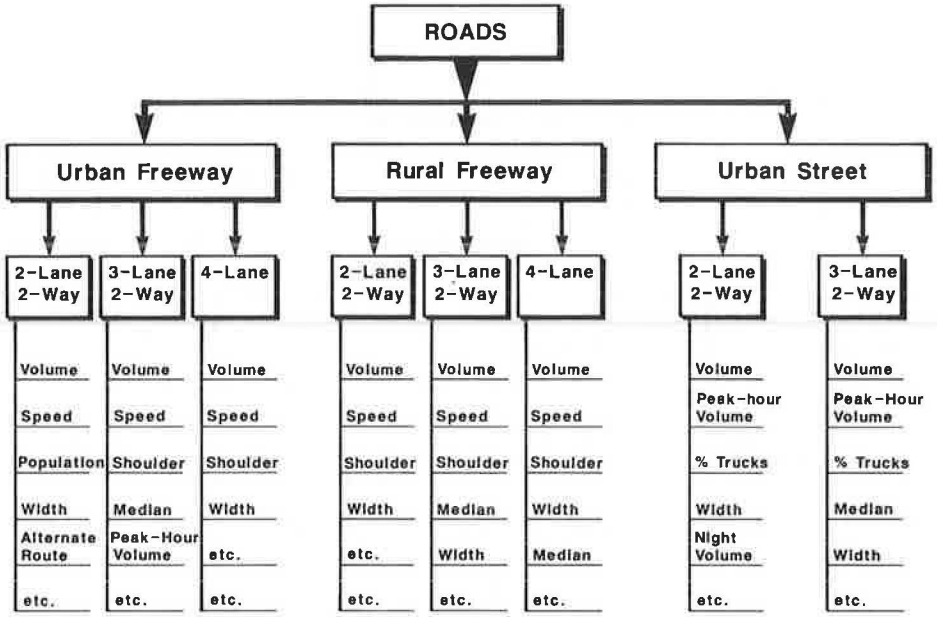

FIGURE 2 Frame network.

Each of the two parts of the antecedent in Rule 3 is called an "expression," or "if clause." The consequent usually contains a single expression, or "then clause"; it could contain more than one. The clauses in the antecedent can be connected by the logical operator "and" or "or."

In a rule-based expert system, knowledge is represented as sets of rules that are checked against a collection of facts or knowledge about the current situation. When the antecedent of a rule is satisfied by the facts, the action specified by the consequent is performed. When this happens, the rule is said to "fire" or "execute" (*1*). A rule interpreter compares the antecedents with the facts and executes the rule whose consequent matches the facts, as follows:

Facts: A flammable liquid was spilled.
The pH of the spill materials is less than 6.
The spill smells like vinegar.
Rule: If the pH of the spill is less than 6, the spill material is acid.

The new fact is added to the knowledge base: The spill material is an acid.

The action of the rule may modify the set of facts in the knowledge base by adding a new fact. The new facts added to the knowledge base can themselves be matched to the antecedent of the rule. The matching of rule antecedents to the facts can produce what are called "inference chains" (*1*). The inference chain for this example is shown in Figure 3. This inference chain shows how the system used the rules to infer the identity of the spill material. An expert system's inference chains can be displayed to the user to help explain how the system reached its conclusions.

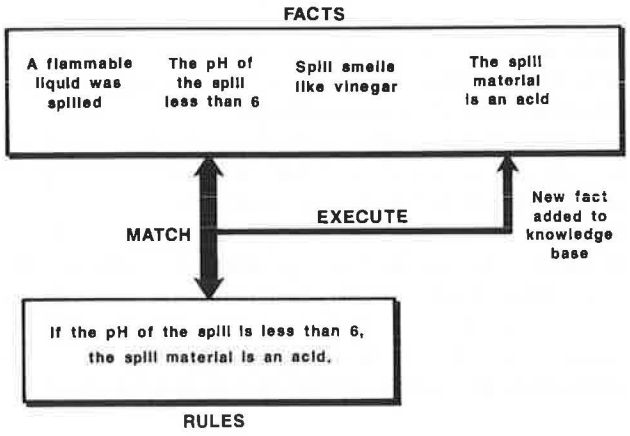

FIGURE 3 Inference chain.

Object-Attribute-Value Triplets

Another way to represent factual information is by object-attribute-value (OAV) triplets. In this scheme, an object may be a physical entity such as a door, a car, or a pavement, or it may be a conceptual entity, such as a logic gate, a bank loan, or a sale. An attribute is a general characteristic or property of an object; for example, interest rate is an attribute of a bank loan. The final member of the triplet is the value, which describes the specific nature of an attribute in a particular situation. For example, the number of lanes on a certain highway might be 6, or the interest rate for a bank might be 12 percent. Figure 4 shows an example of OAV representation.

Representing knowledge with OAV triplets is a specialized form of semantic network. Exotic links are banished in favor of two simple relationships. The object-attribute

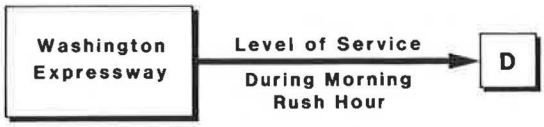

FIGURE 4 OAV representation.

link is a has-a link, and the attribute-value link is an isa link. For example, a bank loan has a rate of interest, and 12 percent is a rate of interest. Nodes are classified as objects, attributes, or values.

EXPERT SYSTEM TOOLS

Expert system tools are the programming languages and support packages used to build the expert system. The three major categories of tools available for expert system building are programming languages, knowledge engineering languages, and system building aids. The most common tools currently used by transportation engineers to develop expert systems are knowledge engineering languages (SHELLs) because they are relatively easy to use. However, depending on the nature of the problem and the kind of representation a knowledge engineer chooses, a different kind of expert system building tool may be employed. The three categories of expert system tools, some of the current commercial systems available in each category, and the kind of representation technique for which each system was designed are described next.

Programming Languages

The programming languages used in expert systems applications are generally either problem-oriented languages, such as FORTRAN and Pascal, or symbol-manipulation languages, such as LISP and Prolog. Currently, the most popular symbol-manipulation language for expert system applications is LISP. A feature of LISP that distinguishes it from most other languages is its mechanism for manipulating symbols. LISP can manipulate symbols readily because of its list structure characteristics. List structures are collections of items enclosed in parentheses, in which each item can be either a symbol or another list. Complex concepts can be represented in and built into an expert system using the list structures.

Problem-oriented languages are generally designed for solving particular classes of problems. FORTRAN, for example, performs algebraic calculations for scientific, mathematical, and statistical problems. Problem-oriented languages have been used in expert system development but are not very popular for extensive applications. Some of the commercial programming languages for developing expert systems are INTERLISTP-d and SMALLTALK-80 (Xerox Corporation); LISP (LISP Machine, Inc.); Prolog (Quintus Computer Systems, Inc.); and GCLISP (Gold Hill Computers, Inc.).

Knowledge Engineering Languages (SHELLs)

Knowledge engineering languages are a subclass of programming languages designed specifically for expert systems development. They fall into two major categories: skeletal systems and general-purpose systems. Removing from the expert system domain-specific knowledge leaves the skeletal system, the inference engine, and the support facilities. Support facilities are the environment associated with a building tool in the expert system that helps the user interact with it. Because skeletal systems apply only to a limited class of problems, they lack generality and flexibility as a building tool method. The structure and built-in facilities of a skeletal system, however, make expert systems development easy and fast. The key decision that must be made initially by the system developer is to select an appropriate SHELL that matches the problem.

In contrast, general-purpose knowledge engineering languages can handle a wide range of problem areas and types. They provide more control over accessing information in the knowledge base than does a skeletal system.

The general-purpose languages, however, may be more difficult to use. Table 1 shows some commercially available SHELLs along with the type of representation technique for which they were designed.

System-Building Aids

System-building aids consist of commercially available software programs that can be classified as either design aids or knowledge acquisition aids. Design aids help the expert system developer design and build an expert system by establishing a framework for the representation of knowledge and its supporting facilities. Knowledge acquisition aids help the expert system builder transfer the knowledge rules and heuristics from the human expert to the knowledge base of an expert system. Available expert system building aids include EXPERT-EASE (Expert Systems International), RULE-MASTER (Radian Corporation), and TIMM (General Research Corporation).

TABLE 1 SELECTED EXPERT SYSTEMS SHELLs

Tool	Representation	Developer
ART	Rule and frame-based	Inference Corporation
DUCK	Logic and rule-based	Smart Systems Technology
EXSYS	Rule-based	Exsys, Inc.
KEE	Rule and frame-based	Intellicorp
M.I.	Rule-based	Teknowledge
OPS 5	Rule-based	Digital Equipment Corporation
S.1	Rule and frame-based	Teknowledge
SRL+	Frame-based	Carnegie Graphics Inc.

CONCLUSION

Presented in this paper are the options available in expert systems technology for building systems to aid in solutions for transportation engineering problems. First, the problem must be defined in terms of the most appropriate way for source knowledge about the problem to be represented. Eight common techniques for representing knowledge are: predicate logic, monotonic logic, statistical and probabilistic reasoning, semantic nets, frames, scripts, rules, and object-attribute-value. Rules are the most commonly used because of the availability of rule-based SHELLs. Second, the problem must be matched to a practical system building tool. When these major decisions are made, the knowledge acquisition and system development process can proceed. The state of the practice of building expert systems will mature when categories of tools can be related to classes of problems that generalize categories of expert systems applications in transportation engineering.

ACKNOWLEDGMENT

This paper was produced with the cooperation of the FHWA, U.S. Department of Transportation.

REFERENCES

1. D. A. Waterman. *A Guide to Expert Systems*. Addison-Wesley Publishing Company, Reading, Pa., 1986.
2. E. C. Carter and W. S. Hamburger. *Introduction to Transportation Engineering*. Reston Publishing Company, Inc., Reston, Va., 1978.
3. J. L. Alty and M. J. Coombs. *Expert Systems*. NCC Publications, Manchester, England, 1984.
4. E. Rich. *Artificial Intelligence*. McGraw-Hill Book Company, New York, 1983.
5. Y. Sheffi. *Urban Transportation Networks*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.