Application of Expert Systems to Left-Turn Signal Treatment

EDMOND CHIN-PING CHANG

Left-turn treatments are essential to signal capacity and operational safety at signalized intersections. Left-turn warrants and guidelines are sets of evaluation procedures designed to maximize level of service, minimize approach delay, and reduce left-turn–related accidents. Currently, three left-turn phasings are used to allow vehicles to make left turns on a green arrow or a circular green indication: permissive, exclusive, and exclusive/permissive. An experimental expert system design for recommending alternative left-turn phase selection on microcomputer systems was investigated. The goal of this study was to computerize left-turn phase selection by using artificial intelligence languages and knowledge engineering. This study focuses on investigating expert systems programming using PROLOG and the INSIGHT 1 system in an IBM PC/XT/AT microcomputer environment. Three experimental systems were developing using the PD PROLOG system, the TURBO PROLOG system, and the INSIGHT 1 knowledge engineering system. The background of the study, the artificial intelligence concept, the basic systems design, and the practical experience gained are discussed. Potential advantages and disadvantages of developing expert systems using different artificial intelligence languages and the knowledge engineering for traffic engineering applications are evaluated. The results of this limited study indicate that it is feasible to combine artificial intelligence and traffic engineering technologies for alternative traffic signal analysis.

This study was developed by the Texas Transportation Institute to investigate the feasibility of applying artificial intelligence (AI) technology and expert systems design concepts to a confined traffic engineering problem using an IBM PC/XT/AT microcomputer. Prototype expert systems were experimented with to analyze user input; evaluate it using various paths of reasoning; offer a conclusion; and, finally, suggest suitable left-turn phase treatment. The guidelines applied in this study were developed from a paper by Jonathan E. Upchurch *(1)*.

Three prototype expert systems were developed with AI programming tools for expert systems using PROLOG and the INSIGHT 1 system in IBM PC/XT/AT-compatible microcomputer systems *(2–4)*. Two slightly different expert systems were designed using AI languages; another system was built with a knowledge engineering tool. These systems include the ones developed in the AI programming languages PD PROLOG and TURBO PROLOG as well as the INSIGHT 1 production rule language *(5–11)*. All three expert systems were completed and observed to perform successfully; advantages and disadvantages were noted for each of the expert system programming techniques.

Texas Transportation Institute, Texas A&M University System, College Station, Tex. 77843.

PD PROLOG is a public-domain experimental PROLOG system that follows very closely the structure and syntax of an AI computer programming language as described by W. F. Clocksin and C. S. Mellish *(5)*. This A.D.A. PROLOG interpreter was developed for educational and public-domain usage *(8)*. TURBO PROLOG is a commercially available AI programming language compiler developed and released in May 1986 by Borland Incorporated *(9)*. It follows more closely the function and syntax of the LISP AI programming language than did the original PROLOG languages, such as the PD PROLOG system. The major advantage of the TURBO PROLOG system is its capability of compiling and generating object codes as quickly as the TURBO PASCAL compiler. It also has built-in editing and tracing functions, a knowledge inquiry environment, knowledge data base management systems, and programming development environments.

INSIGHT 1, as mentioned previously, is a commercially available knowledge engineering tool developed by Level Five Research *(10, 11)*. It was used in this study to investigate the feasibility of designing expert systems using knowledge engineering tools in an IBM PC/XT/AT-based microcomputer environment. In general, AI programming can be implemented through the LISP- or PROLOG-based language system with a minimum of difficulty. Knowledge engineering tools like the INSIGHT 1 system can allow noncomputer-oriented users and knowledge engineers to prototype a specialized problem area quickly. Knowledge engineering tools can assist users to develop their own customized expert system applications and define the logical reasoning structure in less time than it would take any other computer-programming language or system.

BACKGROUND

Left-turn treatments are essential to signal capacity and operational safety at an intersection. Left-turn guidelines are sets of procedures designed to maximize level of service, minimize approach delay, and reduce left-turn–related accidents. Three left-turn phasings are commonly used to allow vehicles to make left turns on a green arrow or circular green indication: permissive, exclusive, and exclusive/permissive left-turn treatments *(1)*. Selecting proper left-turn phasings involves a series of engineering decisions instead of an algorithmic process. The experience and knowledge of a traffic engineer can greatly improve final solutions. The design process begins with describing the intersection geometry, traffic movements, and available signal control equipment. Next, traffic volume data are investigated. When enough information has been collected,

the traffic engineer can propose alternatives. Then the traffic engineer can modify or insert new production rules based on his design experience.

This left-turn phase expert system follows the guidelines recommended by Upchurch *(1)*. Many preidentified factors and rules are required to determine the logical choices among different design alternatives. The evaluation guidelines, as shown in Figure 1, recommend different phasing selections by considering left-turn volumes, opposing through volumes, number of opposing lanes, cycle length, approach speed, sight distance restrictions, and historical records of severe left-turn–related accidents. This selection guideline represents the typical analysis process of *(a)* an algorithmic method, *(b)* knowledge inference capabilities, and *(c)* the knowledge base of a traffic engineer. The first evaluation determines the critical volume cross-product calculation from the input. The forward-chaining inference mechanism models the dependencies among different decision-making activities in the human reasoning process. The reasoning or inference process optimizes design objectives by starting from known information. The third process models the domain knowledge in IF-AND-THEN-ELSE rules to resemble the human decision-making process. For example, the existence of sight distance restrictions and severe left-turn accidents can justify the provision of protected left-turn signal treatments.

These decision rules and reasoning processes are particularly useful for solving problems in instances that may not be covered by established guidelines. Problem-solving expert systems based on established guidelines can provide users with reasoning knowledge similar to that of a human expert constantly

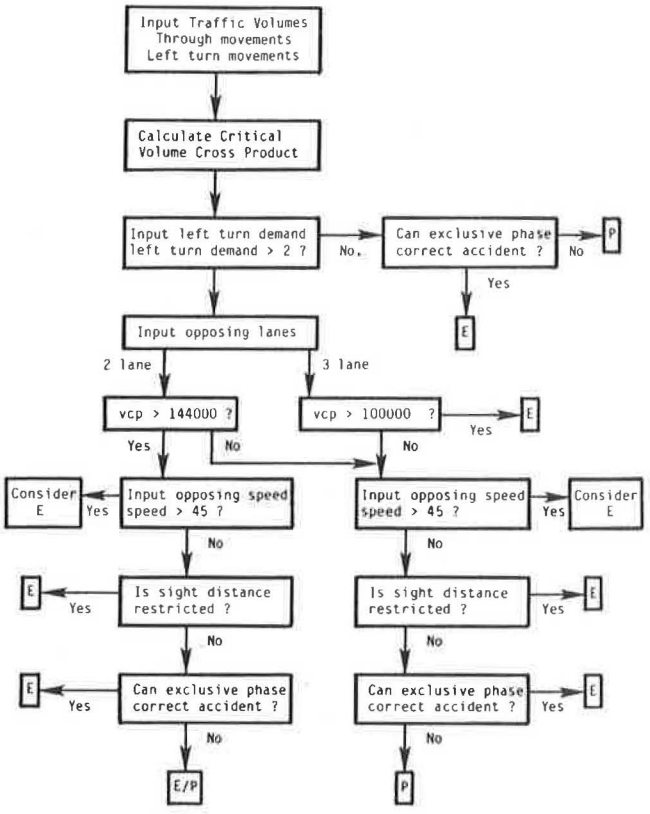

FIGURE 1 Recommended guidelines for selecting type of left-turn phasing *(1)*.

available for assistance in the specialized area *(3, 4)*. The expert system can generate solutions that resemble the traditional design and that may be used by other traffic engineers for determining proper traffic control. Because only a few heuristic decisions that might lead to the best solutions are selectively analyzed each time, the system is quite efficient. Most traffic engineering problems have characteristics similar to left-turn phasing selection as described in this paper. Traffic engineering expert systems are useful for assisting users to solve recurring design problems, sharing common working experience for mutual learning, and providing better design alternatives in the future. By correctly constructing the knowledge-based expert system, traffic engineers can further refine their mental decision-making process to reflect experience obtained from the previous design process.

ARTIFICIAL INTELLIGENCE TECHNOLOGY

Artificial intelligence (AI) technology, including knowledge-based systems and expert systems, has promising applicability to engineering problems *(2–4)*. The relationships among AI, knowledge-based expert systems (KBESs), and expert system (ES) design are shown in Figure 2. Since World War II, scientists have developed computerized techniques to simulate human behavior and decision making. Behavioral scientists, mechanical engineers, and computer scientists are all active in AI research to produce programs that can solve problems that humans solve well. It is anticipated that the AI study will eventually lead to intelligent computerized applications in specialized areas. The research includes decision-making systems, robotic devices, and various approaches to computerized speech synthesizing. Today, the United States, Japan, Britain, and other countries of the European Economic Community are all implementing knowledge-based systems and expert systems. However, expert systems research in this country is confined to only a few university research laboratories, mainly those at Stanford, Carnegie-Mellon, and the Massachusetts Institute of Technology.

Knowledge-Based Expert Systems

The knowledge-based expert system (KBES) is a collection of AI techniques and analysis processes that enables a computer to assist people in analyzing specialized problems. KBESs were introduced to extend computer applications. A KBES

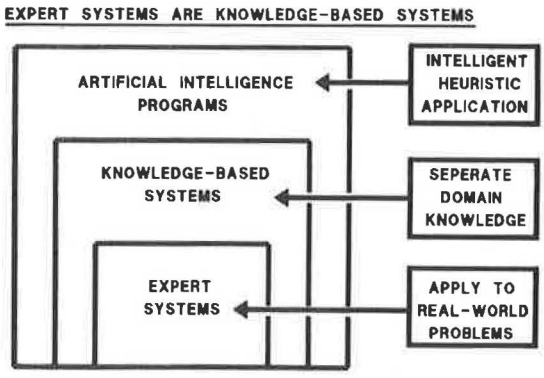

FIGURE 2 Artificial intelligence, knowledge system, and expert systems *(3)*.

provides human expertise through both the knowledge engineering language and the program-supporting environment *(3)*. The AI/KBES application requires development of a generalized knowledge base that permits traffic engineers to interact with the following three components: the traffic characteristic data, the theoretical or simulation results, and the specific hypothesis for measuring the effects of traffic control system measures. The structured guidelines for traffic engineering problems are suitable for KBES applications because explicit algorithms do not exist and the traditional programs can provide only restricted problem-solving capability. A rule-based expert system (RBES) is another knowledge-based mechanism available for design applications. It should be noted that a KBES may also be an RBES.

Expert Systems Design

Expert systems (ESs), as part of the AI/KBES technology, are computer programs that incorporate the knowledge and thinking processes of experts to provide operational people the insights gained from years of experience. Expert systems differ from conventional data-processing programs. The latter rely on defining logical algorithms for a program. The major differences among ESs are expert performance, symbolic reasoning, depth of knowledge representation, and self-knowledge for logical operation. Traditional programs are developed by explicitly stating all of the applicable rules and execution sequences. Usually, algorithmic programming states only the action parts of the rules. A KBES, on the other hand, uses the same action rules as algorithmic programs but specifies independently all of the heuristic parts of the selection sequence. The rules can be programmed in symbolic relationships and treated as the knowledge base.

A practical ES includes three elements: the knowledge data base, the support environment, and the end user. These are usually specified by a knowledge engineer or AI programmer who specializes in ES and a domain expert who understands the specific problem or domain area of the designated program. By conducting extensive interviews with the domain expert, the knowledge engineer can summarize the expert's knowledge into commonly known facts and rule-of-thumb tricks that the expert has acquired from years of experience. Three modules are generally programmed in ES: the explanation module, the knowledge acquisition module, and the user interface module. The explanation module provides the ES with the ability to recommend problem-solving strategies based on the reasoning process. The knowledge acquisition module coded in the knowledge base is usually constructed in rigid format for logical processing. The knowledge interface module often uses a set of problem-oriented questions presented through a friendly interface. The interface module helps the user to monitor system performance, supply information, request explanations, and redirect possible recommendations.

The ES design combines the decision-making process and rule-of-thumb guidelines for specific problem solving. This design process combines the algorithmic method, knowledge inference capabilities, and the knowledge base of the traffic engineer. Sequential control is used to evaluate the critical volume cross-product from input traffic. The forward-chaining concept evaluates the dependencies among different activities in the human reasoning process. This reasoning process optimizes objectives by starting from known information. In the decision-making process, the domain knowledge is written with IF-THEN-ELSE rules to resemble the human decision-making process. For example, the existence of sight distance restrictions and severe left-turn accidents may justify the use of a protected left-turn signal treatment, which might also be recommended by an experienced traffic engineer. Expert systems have been applied in many disciplines. However, not all areas are suitable for expert system formulation *(6)*.

Other Representation Frameworks

Representing knowledge in an AI program means choosing a set of conventions and structures for describing the objects, relations, and processes *(4)*. First, a conceptual framework is chosen to represent the problem, either symbolically or numerically. Then conventions within given computer languages are chosen for implementing the design. The former is difficult and important; the latter is less difficult and of less importance because good programmers can find ways of working with almost any concept within any kind of programming language. Representing knowledge in procedures is one alternative that domain experts in every scientific field have tried hard to avoid. The definition of production rules offers opportunity for making a knowledge base easier to understand and modify.

Artificial Intelligence Languages

A knowledge engineer converts an expert's knowledge into rules that a computer understands. Most programming is done in high-level languages, such as BASIC, COBOL, FORTRAN, PASCAL, and C. AI languages are useful in designing an ES. They include *(a)* high-level AI conventional languages, *(b)* knowledge engineering development tools, and *(c)* portability among different operating systems. Figure 3 shows AI language development *(2)*. AI researchers have been developing LISP machines that can run the knowledge systems more efficiently than does conventional hardware using a standard operating system. If portability is the primary concern, the researchers will choose to translate their codes into conventional languages that can be run on conventional operational systems. On the other hand, if more sophisticated ESs are needed, the tools may be coded for LISP- or PROLOG-based machines.

Currently, several AI languages are available for building expert systems. Specifically, an ES may be implemented as part of the KBES using a general-purpose programming language, general-purpose representation language, or domain-independent expert system framework. These high-level AI languages contain some special features, such as developing reasoning strategies. AI languages contain powerful abstract mechanisms that make the programming of human reasoning logic flexible and easy. Currently, KBESs built using LISP and PROLOG are popular among researchers. ES development tools or knowledge engineering tools can compile these English-like rules into an efficient machine code for developing production expert systems.

The AI programming language normally used is LISP (LISt Processing). LISP is preferred by AI engineers in the United

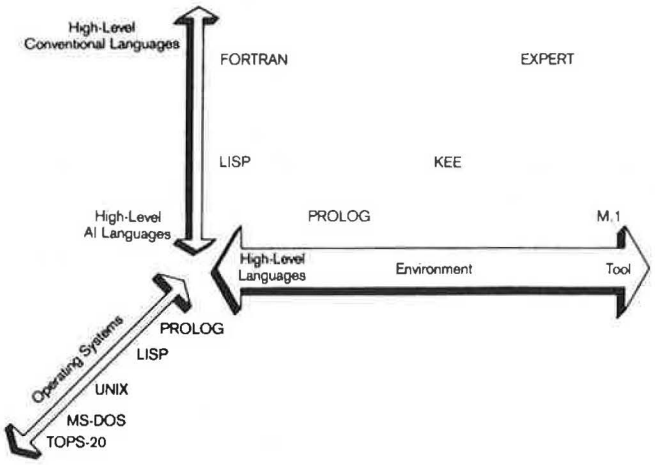

FIGURE 3 Development of artificial intelligence languages *(3).*

States whereas a similar language, PROLOG (PROgramming in LOGic), is preferred in Europe and Japan. PROLOG contains structures suitable for programs that manipulate logical expression, whereas LISP contains operators that facilitate the creation of programs that manipulate lists. These structures are useful for developing symbolic computing programs for numerical programming. LISP has been used for AI programming and ES design for nearly two decades *(3, 4)*. It is a symbolic manipulation language with structures based on the number of constructs or statements. LISP was created by John McCarthy in 1958. Of all the major programming languages still in use, only FORTRAN is older. LISP is highly recursive, and both data and programs can be represented as lists. The lists can be nested like a Chinese "puzzle box." LISP is a "function application" language that uses a set of simple functions, such as (Plus 2 2). PROLOG, in contrast, was initially developed as a symbolic programming language in 1972 by A. Colmerauer and P. Roussel at the University of Marseilles *(5, 8, 9)*. This programming language implements a simplified predicate calculus as a true logical language. PROLOG is a "declarative query" AI language that uses simple relations among fact, rule, and query.

Neither LISP nor PROLOG is an algorithm or a procedure to be executed in fixed sequence, but both program languages can represent the human inference process. In general, AI languages are more flexible and difficult to use in prototype expert system development. Knowledge engineering tools can help an engineer design an ES.

DESIGN PROCESS

Evaluation and design procedures are needed to develop a practical expert system application. The basic design process used to implement this left-turn phase selection ES is shown in Figure 4. The design process implemented includes six different steps: extracting basic information, defining the determining factors, defining the goals and objectives, determining the analysis constraints, developing the program, and finishing program documentation.

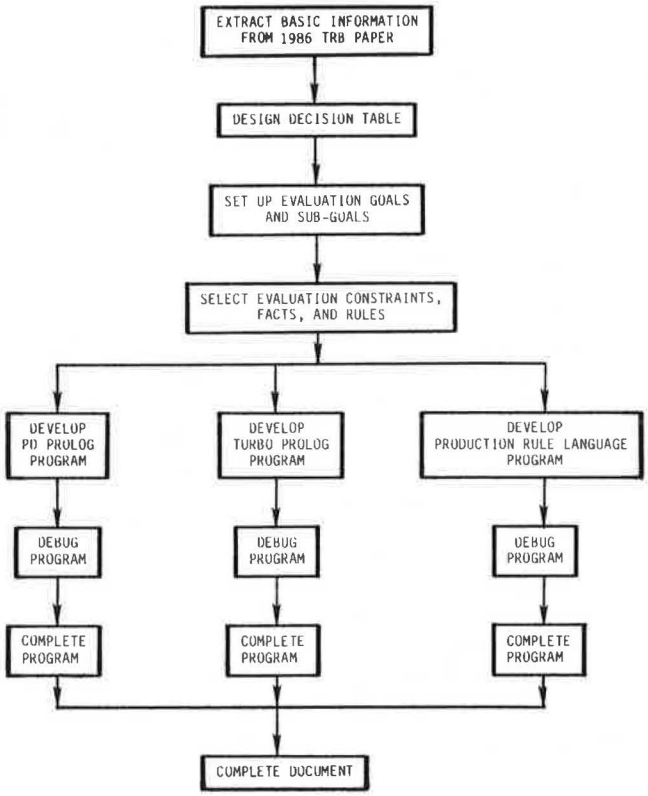

FIGURE 4 Basic design process.

Basic Design Process

Extracting Basic Information

The first step is to describe the characteristics of the problem. The description was first extracted from the design procedure recommended by Upchurch *(1)*. This information contains the analytical format for optimizing the selection of left-turn signal treatments. Basically, it can be classified into main goals, subgoals, facts, rules, and constraints. This basic functional relationship is used in the later analysis. The functional relationship of this particular ES application can be illustrated as follows:

- Study objective: Provide computerized left-turn phase selection process.
- Main goal: Recommend left-turn treatments
  1. Permissive left-turn phase,
  2. Exclusive left-turn phase, and
  3. Exclusive/permissive left-turn phase.
- Subgoal: Different variations of the main goals.
- Fact: Part of the preselected necessary condition.
- Rule: Set of prerequisite conditions to describe each of the subgoals as defined earlier.
- Constraint: Each of the selected prerequisite conditions.

Defining Determining Factors

On the basis of the guidelines recommended for optimizing the left-turn signal treatment process, a decision table was extracted to study the relationships among all of the goals, subgoals, facts, rules, and constraints. As indicated in Figure 1, there are six major constraints or user inputs needed in the decision-making process to determine proper left-turn signal treatments. The constraints, as defined, are the common traffic input information. These user inputs include amount of left-turn demand, number of opposing through lanes, volume cross-product of the conflicting left-turn and through movement pairs, opposing travel speed, sight distance restriction, and possibility of severe left-turn accidents.

Defining Goals and Objectives

From the evaluation procedure recommended in Upchurch's paper, the decision-making process was simplified into three main goals to illustrate this important study result of left-turn treatments. Because the major purpose of this ES is to computerize left-turn phase selection, the study objective is to recommend the phase sequence to be used on the basis of user input. As recommended in Upchurch's guidelines, the goals of this ES design are to recommend exclusive phase, permissive phase, or exclusive/permissive phase treatment as described earlier. To clarify the basic relationship among the outcomes of different data input, these main goals were further divided into 16 different subgoals to accommodate various possible cases involved in the logical design. These conditions or subgoals were separately described as Conditions A through P, depending on their probabilities of occurrence.

Defining Analysis Constraints

The basic analysis constraints are the major factors that can be used to define and describe the goal and subgoal needed in the analysis. Design constraints, facts, and rules were then evaluated according to decision table analysis. The constraints used in this ES include the following user-input variables:

1. Amount of left-turn demand greater than or less than two per cycle,
2. Number of opposing lanes equal to two or three,
3. Volume of cross-product value,
4. Opposing speed greater than or less than 45 mph,
5. Sight distance with or without restriction, and
6. Existence of severe left-turn accidents.

As indicated, some of the questions require numerical data input and other questions need logical data input for expanding the constraints as well as the answer for the object-answer or the simple-facts type of query. Each subgoal or condition is described by the preselected conditions from the user input. Their probability of occurrence depends on the fulfillment of each preselected condition in the actual execution. It should also be noted that the execution sequence of the ES is not predefined but results from evaluating the user's input information.

Developing the Program

After the program was designed, the computer codes were developed using PD PROLOG, TURBO PROLOG, and the INSIGHT 1 production rule languages. It should be noted that these programming tools were selected to implement this ES in the microcomputer environment. The left-turn signal phase selection expert systems were programmed for each goal and subgoal by using the constraints, rules, and facts defined previously. For each expert system, a prototype program was developed, coded, and debugged, and the basic program code was completed. As indicated earlier, AI/ES programming is different from conventional programming because greater emphasis is placed on description of the solution itself than on the solution process. Because most AI/ES programming tools are equipped with a programming-support environment, program development can be completed efficiently.

Finishing Program Documentation

All of the necessary program documentation was implemented inside the ES program to provide information for each of the expert systems. It was also noted that the structured syntax and program code of the AI languages and knowledge engineering tools was useful for internal program documentation.

Decision Table Analysis

A decision table can assist in the evaluation of the major study factors and their corresponding relationships as the different goals, subgoals, and constraints apply in the design process. Table 1 is a simplified decision table to illustrate how an action is represented in the evaluation. In this case, if one of the conditions is not satisfied, no action will be recommended. As

TABLE 1 EXAMPLE OF A SIMPLE DECISION TABLE

CONDITIONS	CHOICE (TRUE OR FALSE)	
LEFT TURN DEMAND > 2 PER CYCLE	TRUE	ELSE
ARE THERE TWO OPPOSING LANES ?	TRUE	ELSE
IS VOLUME CROSS PRODUCT > 100,000	TRUE	ELSE
ACTIONS		
1. SUGGEST USING EXCLUSIVE PHASE	X	
2. CHECK OTHER INPUT VARIABLE		X

TABLE 2 DECISION TABLE DESIGN

LEFT TURN SIGNAL TREATMENTS	PERMISSIVE PHASE			EXCLUSIVE PHASE											E/P PHASE
CONDITIONS	A	O	P	C	D	E	G	H	I	J	K	L	M	B	F
LEFT TURN DEMAND															
o DEMAND > 2		X	X	X	X	X	X	X	X	X	X	X	X		X
o DEMAND <= 2	X													X	
OPPOSING THROUGH LANES															
o OPPOSING LANES = 2		X		X	X	X	X	X	X						X
o OPPOSING LANES = 3			X							X	X	X	X		
VOLUME CROSS PRODUCT															
o > 144,000				X	X	X									X
o <= 144,000		X					X	X	X						
o > 100,000													X		
o <= 100,000			X							X	X	X			
OPPOSING SPEED															
o > 45				X			X			X					
o <= 45		X	X		X	X		X	X		X	X			X
SIGHT DISTANCE															
o W/ RESTRICTION					X			X			X				
o NO RESTRICTION		X	X			X			X			X			X
SEVERE LEFT ACCIDENT															
o COULD BE CORRECTED BY EXCLUSIVE PHASE						X			X			X		X	
o COULD NOT BE CORRECTED BY EXCLUSIVE PHASE	X	X	X												X

indicated, satisfying only one condition in Table 1 may indicate that other input data are needed in the decision process.

Table 2 is the detailed decision table used in this prototype expert system. The vertical column of the decision table lists all of the major decision factors and their constraints. In this particular example, there are six major determining factors. These factors include left-turn demand > 2 or < 1, opposing lanes = 2 or = 3, volume cross-product > 144,000 or ≤ 144,000 or > 100,000 or ≤ 100,000, opposing speed > 45 or ≤ 45 mph, sight distance with restriction or no restriction, and the possibility of correcting severe left-turn accidents by exclusive phase.

In the decision table, the first horizontal row lists all of the main goals (i.e., permissive phase, exclusive phase, and exclusive/permissive phase). The second horizontal row lists all of the possible subgoals ranging from Conditions A through P. In each column, X represents the requirements for fulfilling a certain decision condition. For example, in Condition A under the selection of the permissive phase, there are two Xs, one representing left-turn demand ≤ 2 and the other the severe left-turn accidents that cannot be corrected by using exclusive left-turn phasing. The existence of these two conditions causes the permitted left-turn phase treatment to be recommended.

This basic decision table structure can also be transformed into the pseudocode shown in Figure 5. Two different mechanisms, using either flowcharts or decision tables, are used to illustrate both decision analysis by domain experts and expert systems programming by knowledge engineers. A program flowchart can help the domain expert trace a path in the program and the knowledge engineer or programmer develop the AI program. On the other hand, the detailed decision table can also be used to identify the requirements for each condition and the desired goals from the available information. By using results from the decision table, an efficient pseudocode for later

```
1.  Define decision rules.
2.  Input 4 arterial NEMA traffic movements: 2, 5, 6, 1.
3.  Return the maximum among the volume cross products of 2 * 1 and 5 * 6.
4.  Input left turn demands and sight distance restriction.
5.  (If (left turn demand <= 2) then
        ("Can exclusive phase correct accidents?" and
        Echo printouts and
        (If can then output "Exclusive phase - suggested".) or
        (If cannot then output "Permissive phase - suggested".)) or

    (Else Input opposing lane and
          (If (the maximum product <= 144000 ; opposing lane = 2) then
              (If (opposing lane = 3, maximum product > 100000) then
                  (Output "Exclusive phase - suggested".) or
              (Else Input opposing speed and
                    (If (opposing speed > 45) then
                        (Output "Exclusive phase - suggested".) or
                    (Else "Is sight distance restricted?" and
                          (If sight distance is restricted then
                              (Echo printouts and
                               Output "Exclusive phase - suggested".) or
                          (Else "Can exclusive phase correct accidents?" and
                                Echo printouts and
                                (If can then
                                    (Output "Exclusive phase - suggested".) or
                                (Else
                                    (Output "Permissive phase -suggested".)
                                ))))))))).

          (Else input opposing speed and
                (If (opposing speed > 45) then
                    (Output "Exclusive phase - suggested".) or
                (Else "Is sight distance restricted?" and
                      (If sight distance restricted then
                          Echo printouts and
                          (Output "Exclusive phase - suggested".) or
                      (Else "Can exclusive phase correct accidents?" and
                            Echo printouts and
                            (If can then
                                (Output "Exclusive phase - suggested".) or
                            (Else
                              (Output "Exclusive/Permissive phase - suggested".)
                            ))))))).
```

FIGURE 5 Program pseudocode.

programming can be developed. In summary, the basic advantages of using the decision table include definition of all constraints individually for each goal and subgoal, presentation of all information clearly and systematically, and provision of more efficient program structure.

Program Structure

As indicated in Figure 4, this logical structure was implemented in three expert systems. They were programmed with the PD PROLOG and TURBO PROLOG computer languages and the INSIGHT 1 knowledge engineering tool *(8–11)*.

PD PROLOG Program

A PD PROLOG program is defined with the IF-THEN-OR-AND-ELSE rule *(8)*. There are two major advantages to this programming language. First, it uses the OR function that can greatly reduce redundant rules in programming. Second, it is quite similar to the pseudocode, as illustrated, for internal program documentation. These advantages make PD PROLOG programs easier to understand. However, this programming approach also has two major disadvantages: *(a)* the program is hard to trace for program execution and *(b)* it cannot trace backward to provide backward-chaining analysis for evaluating a specific subgoal in this ES design.

TURBO PROLOG Program

A TURBO PROLOG program consists of four basic programming blocks that include definitions of the domains, predicates, goals, and clauses *(9)*. The domain and predicate blocks identify all of the variables, types, and functions. The goal block declares the desired destinations or recommendations of searching. The clause block properly defines all of the facts, rules, functions, and procedures. The major advantages of TURBO PROLOG are that it is easy to understand, easy to debug, suitable for modular programming, able to generate execution files, linkable with other language programs, and equipped with editing and tracing functions. The major drawback to TURBO PROLOG is its incapability of using the OR function in the production rule. This drawback means duplicate production rules are needed to define specific conditions for each similar alternative in the expert systems design.

INSIGHT 1 Production Rule Language

INSIGHT 1 programs contain two basic parts *(10, 11)*. The first declares the goals and subgoals; the second defines all of the rules and facts. In this study, the first part of the INSIGHT 1 program describes the goals and subgoals of the left-turn signal treatments, such as different left-turn treatments and Conditions A through P. The second part of the INSIGHT 1 program summarizes the interrelationships of the prerequisite rules for fulfilling the conditions in the evaluation process.

There are two ways to program an INSIGHT 1 rule with knowledge engineering tools. The first is to separate the programming into two separate but coordinated modules. One module deals with the definition of the goals and subgoals. The other module declares the individual rules, facts, and functions for each definable case in the decision-making process. In this approach, the user can search for each individual goal or subgoal as a separate entity. The second approach mixes the goals with the facts and rules. That is, the main goals or subgoals are defined again as separate facts and rules to be included in the decision evaluation process. In either case, the user is required to supply only the necessary information related to the query; the expert system will search for certain main goals and decide which alternative is most suitable according to the user's choice for generating the optimum solution. The user may prefer the second approach that mixes the goals and rules. However, from the programmer's point of view, the first approach of separating the goals and rules is much easier to use to develop and debug the INSIGHT 1 program codes.

CONCLUSIONS AND RECOMMENDATIONS

This study investigates the feasibility of applying AI technology to the development of a prototype expert system in transportation engineering for microcomputer application. The basic procedure for generalized expert systems design generated from established guidelines has been summarized. Three slightly different expert systems were developed using the PD PROLOG and TURBO PROLOG languages and the INSIGHT 1 knowledge engineering tool. Table 3 gives a comparison of the advantages and disadvantages of the three prototype expert systems. In this table is summarized some of the design experience gained from programming this simplified traffic engineering analysis for application in the IBM PC/XT/AT microcomputer environment. This investigation is focused on knowledge acquisition, knowledge representation, system programming, and future applications. However, it is believed that this prototype expert system still requires some improvements before any practical applications can be made.

Conclusions

The following conclusions were reached in the course of designing the prototype system to optimize left-turn signal analysis:

1. Expert systems are appropriate for preidentified problem solving,
2. AI languages and knowledge-based engineering tools have advantages and disadvantages, and
3. The ES programming approach may be tailored for practical applications.

It was concluded that knowledge engineering tools, such as INSIGHT 1, do indeed have some advantages over conventional AI languages. The major advantages are their easy-to-read-and-write programs, the user-friendly menus supported by the programming environment, and the clearly defined goals and subgoals. Built-in functions are available for explaining questions in the knowledge engineering programming environment. For program debugging, a trace report is provided to study program execution and the knowledge inference process. The program-supported windows and functions are also useful for easy program development and operation. Most of all, both

TABLE 3 COMPARISONS OF TURBO PROLOG, PD PROLOG, AND INSIGHT 1

	TURBO PROLOG	PD PROLOG	INSIGHT 1
DECLARATION OF VARIABLES, OBJECTS.	YES (DOMAINS)	NO	NO
DECLARATION OF FUNCTIONS OR RELATIONS.	YES (PREDICATES)	NO	NO
DECLARE THE SEARCHING ROUTE	YES (GOAL)	NO	YES (GOALS & SUBGOALS)
SET UP RULES, FACTS, & FUNCTIONS.	DEFINE INSIDE CLAUSES BLOCK	DEFINE INSIDE THE PROGRAM	USE RULE FUNCTION TO DO THE JOB
READABILITY	VERY EASY	EASY	VERY EASY
PROGRAMMING DIFFICULTY	EASY	EASY	VERY EASY
CAN GENERATE .OBJ FILE .EXE FILE	YES	NO	NO
INTERPRETER MODE	YES	YES	YES
TRACE FUNCTION	YES	NO	YES. (GENERATE TRACE REPORT)
STRUCTURED PROGRAM	YES	YES	YES
USE OF PARENTHESE	NO	YES	NO
USE OF OR FUNCTION	NO	YES	NO
CAN LINK WITH OTHER COMPUTER PROGRAM	YES	NO	NO
ADD EXPLANATION FOR EACH GOALS & SUBGOALS	BY PRINT COMMAND NEED TO DIFINE FORMAT	BY PRINT COMMAND NEED TO DIFINE FORMAT	BY EXPAND FUNCTION VERY HANDY TO USE
DEBUGGING DIFFICULTY	VERY EASY	NOT EASY	EASY
SELF PROVIDE MENU, WINDOW, & FUNCTION KEYS	NO NEED DEFINED BY PROGRAMMER	NO NEED DEFINED BY PROGRAMMER	YES
DO MATH OPERATION	YES	YES	NO
WORK FORWARD & BACKWARD	NO NEED TO MODIFY THE PROGRAM	NO NEED TO MODIFY THE PROGRAM	YES
BUILT-IN EDITIOR	YES	NO	YES

the forward- and backward-chaining capabilities are available in the INSIGHT 1 system to *(a)* check whether goals or subgoals fit the input (forward chaining) and *(b)* check the input for the fulfillment of specific goals or subgoals in the analysis (backward chaining).

Recommendations

It is recommended that future expansions and improvements be made to develop large-scale expert systems for practical traffic management and engineering applications. Four points were ignored in this prototype expert systems design. These were the abilities to

1. Provide input data checking in the query process,
2. Return to previous steps to make changes during the search process,
3. Abandon the current searching process without losing input data, and
4. Change the data base at any time.

At present, AI researchers are trying to develop large-scale expert systems for production usage. They have devoted a lot of effort to making the expert systems design more flexible and understandable for general applications. The stylized condition-action knowledge representation provides many advantages because of its simplicity and restricted syntax in the natural language interface mechanism. Similarly, explanations and reasoning are also simplified because the convenient backward-chaining structure dynamically links the knowledge rules for logical reasoning from both directions.

Because expert systems represent a relatively new technology, there are two challenges that face most AI/ES applications today. One is how to define and represent knowledge for intelligent application by computers. The other is to develop better ways to use expert knowledge for intelligent problem solving. Although new knowledge representations in AI/ES designs have not been fully developed, AI experts are still experimenting with various knowledge representations in different discipline areas. Specialists in many fields are encountering difficulty in encoding field expert knowledge. Many refinements to the design process are still needed to satisfy demands for expert problem solving. These efforts include the modification of production rules for future reasoning in a detailed and ill-structured domain environment.

An expert system is best suited to applications in which the subject is highly detailed but tightly defined, such as practical traffic engineering applications. In the AI/ES design, the

requirements of each goal or subgoal may be displayed to allow the user to understand the decision-making process. Because both forward- and backward-chaining capabilities are provided, users can refine their expertise by evaluating in both directions to search for a specified goal or subgoal. The AI/ES approach has a dual function. On the one hand, it provides an approach to computerizing decision analysis based on expert knowledge. On the other hand, AI/ES development presents a new opportunity for domain experts in many fields to review their expertise systematically through AI techniques. Therefore the development of specialized problem-solving tools can also contribute to refinement of the reasoning logic of particular applications.

A recommended AI/ES approach is shown in Figure 6. As indicated, the user's decision-making process can be used to construct the basic decision table. Then sets of knowledge engineering tools can be used to refine the logical reasoning from the knowledge representation and acquisition process during the construction of the production expert system. If more user-oriented applications are needed in the future, the expert system can later be translated into programs using AI languages for fast execution. In this way, the domain expert and user may cooperate more quickly to develop productive expert systems for their specialized applications.

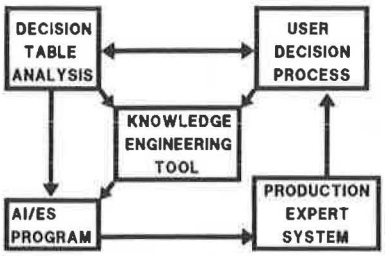

FIGURE 6 Recommended AI/ES programming approach.

ACKNOWLEDGMENTS

The author appreciates the research support of the Texas Department of Highways and Public Transportation in cooperation with the Federal Highway Administration, U.S. Department of Transportation, and the Texas Transportation Institute. In addition, the constructive comments and suggestions provided by the TRB reviewer were helpful during the completion of this paper.

REFERENCES

1. J. E. Upchurch. Guidelines for Selecting Type of Left Turn Phasing. Presented at 65th Annual Meeting of the Transportation Research Board, Washington, D.C., Jan. 1986.
2. P. H. Winston. *Artificial Intelligence*, 2nd ed. Addison-Wesley, Inc., Reading, Mass., 1984.
3. P. Harmon and D. King. *Expert Systems: Artificial Intelligence in Business*. John Wiley and Sons, Inc., New York, 1985.
4. B. G. Buchanan and E. H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Inc., Reading, Mass., 1985.
5. W. F. Clocksin and C. S. Mellish. *Programming in PROLOG*, 2nd ed. Springer-Verlag, New York, 1984.
6. C. T. Hendrickson, D. R. Rehak, and S. J. Fenves. Expert Systems in Transportation Systems Engineering. Submitted to *Transportation Research*, 1986.
7. C. Zozaya-Gorostiza and C. T. Hendrickson. An Expert System for Traffic Signal Setting Assistance. Submitted to *Journal of Transportation Engineering*, ASCE, 1986.
8. *PD PROLOG—User's Manual*. PROLOG documentation for the educational and public-domain system. Robert Morein and Automata Design Associates, Dresher, Pa., 1986.
9. *TURBO PROLOG—The Natural Language of Artificial Intelligence—Owner's Manual*. Borland International, Inc., Scotts Valley, Calif., May 1986.
10. *INSIGHT 1 Reference Manual*. Level Five Research, Inc., Indialantic, Fla., 1986.
11. *INSIGHT 2+ Reference Manual*. Level Five Research, Inc., Indialantic, Fla., 1986.

The products mentioned in this paper are trademarks of several companies. IBM Personal Computer (PC) and PC DOS are products of the IBM Corporation. MS and MS DOS are registered trademarks of the Microsoft Corporation. PD PROLOG is a trademark of the Robert Morein and Automata Design Associates. TURBO PROLOG is a trademark of Borland International, Incorporated. INSIGHT 1 is a trademark of Level Five Research, Incorporated.