Role of Expert Systems in Construction Robotics[1]

Steven J. Fenves[2] and Daniel R. Rehak[3]

1. Introduction

Knowledge based expert systems (KBES) are computer programs, based on artificial intelligence (AI) techniques, designed to reach the level of performance of a human expert in some specialized problem solving domain. Expert systems have a great potential for practical use in ill-structured problem solving domains where explicit algorithms do not exist or where traditional computer programs provide only restricted problem solving capabilities.

The emerging area of construction robotics provides a unique opportunity for the application of expert system. Potential expert systems are foreseen at three levels:

- as integral components of construction robots, providing components of the robot's sensing, planning and control functions;
- as support functions to construction robots in the areas of interpretation of site conditions and evaluation of completed tasks; and
- as part of an integrated design process which explicitly takes into account the capabilities and constraints of construction robots.

As the construction process and the construction site both increase in complexity, expert systems will increasingly augment or replace conventional algorithmic programs in which all decisions have to be anticipated and pre-programmed. Expert systems are applicable to a wide range of tasks, and many problems in the construction domain, such as estimating, scheduling, structure diagnosis and site investigation are potential expert system applications. The development of construction robots provides many new applications for expert systems including tasks such as sensor interpretation and equipment diagnosis, operational planning and operational monitoring.

2. Expert Systems Versus Algorithmic Applications

In order to put expert systems into proper context, the nature and limitations of conventional computer programs must first be examined. A computer program consists of a recipe of rules that completely specifies the problem solving sequence and actions:

---

[1]Prepared for presentation at the *Workshop Conference on Robotics in Construction*, Carnegie-Mellon University, Pittsburgh, PA, June 17-20, 1984.

[2]University Professor, Department of Civil Engineering Carnegie-Mellon University, Pittsburgh, PA 15213

[3]Assistant Professor, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA 15213, ARPAnet: REHAK@CMU-RI-CIVE

IF **condition**$_1$ *THEN* **action**$_1$
IF **condition**$_2$ *THEN* **action**$_2$
⋮ ⋮
⋮ ⋮
IF **condition**$_i$ *THEN* **action**$_i$
⋮ ⋮
⋮ ⋮
IF **condition**$_n$ *THEN* **action**$_n$

Each rule contains a premise or condition which is evaluated. If the condition is true, the corresponding action is performed and execution continues with the next rule. If the condition is false, the action is skipped.

The basic model of any program is a pair of rules

IF `input is available` *THEN* `compute results`
IF `results are computed` *THEN* `output results`

A process such as **`compute results`** is further subdivided into thousands of more detailed rules. As illustrated, the rules are intertwined; the action result of one rule becomes the premise of another rule. Traditional computer programs are developed by explicitly stating all applicable rules and their precise sequence of execution. Such programs are called *algorithmic.*

Only a person knowledgeable in the domain of the program, a *domain expert*, can define the applicable conditions and corresponding actions. This is particularly true in any practical engineering program, where a very large proportion of the rules are not necessarily based on the *causality* of physical laws, but represent the *heuristics* (assumptions, limitations, rules of thumb or *style*) of the expert or his organization. Developing a complete set of rules for any engineering application is a major undertaking. The rules must satisfy the following criteria:

- *Completeness.* The set of rules must provide an action for every possible combination of conditions. Often, many of the conditions are not explicitly stated. They may be "second nature" to the programmer or user, or the combination of conditions represents a case which is unusual, and thus not expected to occur. However, failure to consider all possible cases results in a program which will not perform in an acceptable manner over the complete range of potential applications.

- *Uniqueness.* The set of rules must provide one and only one unique outcome for every possible combination of conditions. As outlined above, this is difficult to achieve, and if not achieved it results in a program which is flawed.

- *Correctness.* The set of rules must provide a correct outcome for all possible conditions. In addition, the sequence of rules must be correct. Misinterpretation problems occur when the program (programmer) invokes the wrong action for a given condition. Either the action is wrong, or the proper action is not associated with the proper premise. An incorrect sequence applies rules at the wrong time.

The program developer has the responsibility to insure that the three criteria listed above are met. Due to the complexity and size of any real program, this goal is almost impossible to attain, and this makes conventional program development expensive. Additional costs are incurred when programs are updated; the program developer must not only modify or add rules, he also must locate the affected rules in the sequence and modify the sequence itself.

Even if the completeness, correctness and uniqueness criteria are met, there are still some problems with traditional programs:

- The program assumes all data has been input and is without error, while in many cases only incomplete and uncertain data are available.
- The program functions as a *black box* with no mechanisms to explain how it arrives at the computed results. The user must refer to external documentation, or the code listing itself, to determine the problem solving approach.
- The program contains only limited mechanisms for controlling the problem solving approach. It solves each problem in only one predefined and preprogrammed way -- an *all or nothing* situation.

Applications based on mathematical models and those requiring intense numerical computations may be conveniently built as algorithmic programs. However many problem solving strategies, such as interpretation or design, are *ill-structured* or *ill-defined* [Simon 81], and are not well suited to the rigid algorithmic format. To date, such applications have not been successfully computerized. The use of expert systems for such applications is discussed in the next two sections.

3. Expert Systems

3.1. Overview

Knowledge based expert systems are designed to overcome the shortcomings of algorithmic computer applications. They may be based on the same type of premise-action rules as algorithmic programs, but the sequence of selecting and applying the rules is not specified a priori. The rules are treated as *knowledge* which is used by the knowledge processing component of the expert system. This knowledge processor determines which rules are applicable in any situation and invokes the corresponding actions. By design, the set of rules need not be complete or unique; the knowledge processor knows how to deal with such circumstances. Expert systems have additional capabilities for selecting rules based on incomplete or uncertain data, and explaining why rules are selected and how they are used.

3.2. Background

Knowledge based expert systems have recently emerged from decades of research in artificial intelligence. They are practical problem solving tools that can reach a level of performance comparable to that of a human expert in some specialized problem solving domain. Rather than being a program with general problem solving knowledge that can be applied to any task, an expert system contains a large body of domain specific knowledge gathered from human experts. This knowledge is used to perform problem solving tasks in a manner similar to experts.

Expert systems have been developed in a number of disciplines, including: medical consultation (*MYCIN*) [Shortliffe 76], hypothesizing molecular structure from mass spectrograms (*DENDRAL*) [Buchanan 69], computer configuration (*R1*) [McDermott 80], mathematical formula manipulation (*MACSYMA*) [Moses 71], oil well logging (*Dipmeter Advisor*) [Davis 81, Smith 83], and mineral exploration (*Prospector*) [Duda 79]. Numerous applications are moving from the research laboratory into production [Duda 83]. Many civil engineering problem domains are candidates for expert system formulation [Sriram 82], and several prototypes are under development.

3.3. Architecture of an Expert System

The principal distinction between expert systems and algorithmic programs lies in the use of knowledge. A traditional algorithmic application is organized into data and program. An expert system separates the program into an explicit *knowledge base* describing the problem solving strategy and a control program or *inference machine* which manipulates the knowledge base. The data portion or *context* describes the problem being solved and the current state of the solution process. Such an approach is denoted *knowledge based* [Nau 83].

A variety of expert system architectures exist. Various domain independent systems have different inference procedures and different knowledge representation schemes, including: production systems [Forgy 81], semantic inference networks [Reboh 81], and frame representations [Fox 82]. More complex blackboard systems, which are based on multiple experts operating at different levels of abstraction, also have been built [Balzer 80, Erman 80, Nii 82]. In the production system formalism for domain knowledge representation, the knowledge is represented directly in terms of IF-THEN rules illustrated above. Other types of expert systems use different knowledge representation formalisms. Some of the problem solving strategies incorporated in expert systems are discussed in Section 3.4.

A schematic of an expert system using the production system formalism (IF-THEN rules) is shown in Figure 1. It is to be emphasized than only the knowledge base is domain specific. All the other components are parts of a general purpose expert system building framework. The components of the expert system are:

- *Knowledge Base.* The knowledge base is the repository for all the knowledge and rules used by the system in problem solving. This information can be divided into two classes: the factual or *causal* knowledge of the application domain, and the *empirical* associations or rules. The knowledge base may also contain long term historical information and facts. All the information in the knowledge base is organized so that it may be effectively utilized by the other components of the system.

- *Context.* The context is also denoted *short term memory* (STM). It contains all of the information which describes the problem currently being solved. The context data may be divided into facts provided by the user and those derived or inferred by the program. The use of an expert system program begins with the user entering some known facts about the problem into the context.

- *Inference Machine.* The inference machine or inference engine is the knowledge processor. It operates on the context, utilizing the rules in the knowledge base to deduce new facts which then can be used for subsequent inferences. The basic operation of a *forward chaining* inference machine, discussed in Section 3.4, is an infinite loop performing three steps:

  1. Examine the premises of rules in the knowledge base and determine which of these currently evaluate to true, based on the current problem data maintained in the context. This step, performed by the change monitor or pattern matcher, yields a set of candidate rules.

  2. Select one of the applicable rules. The rule is chosen by the scheduler or processor.

  3. Invoke or *fire* the corresponding action, which will change some items in the context. The context is updated by the knowledge modifier.

The objective of the inference machine is to arrive at a global conclusion (*goal*), and the

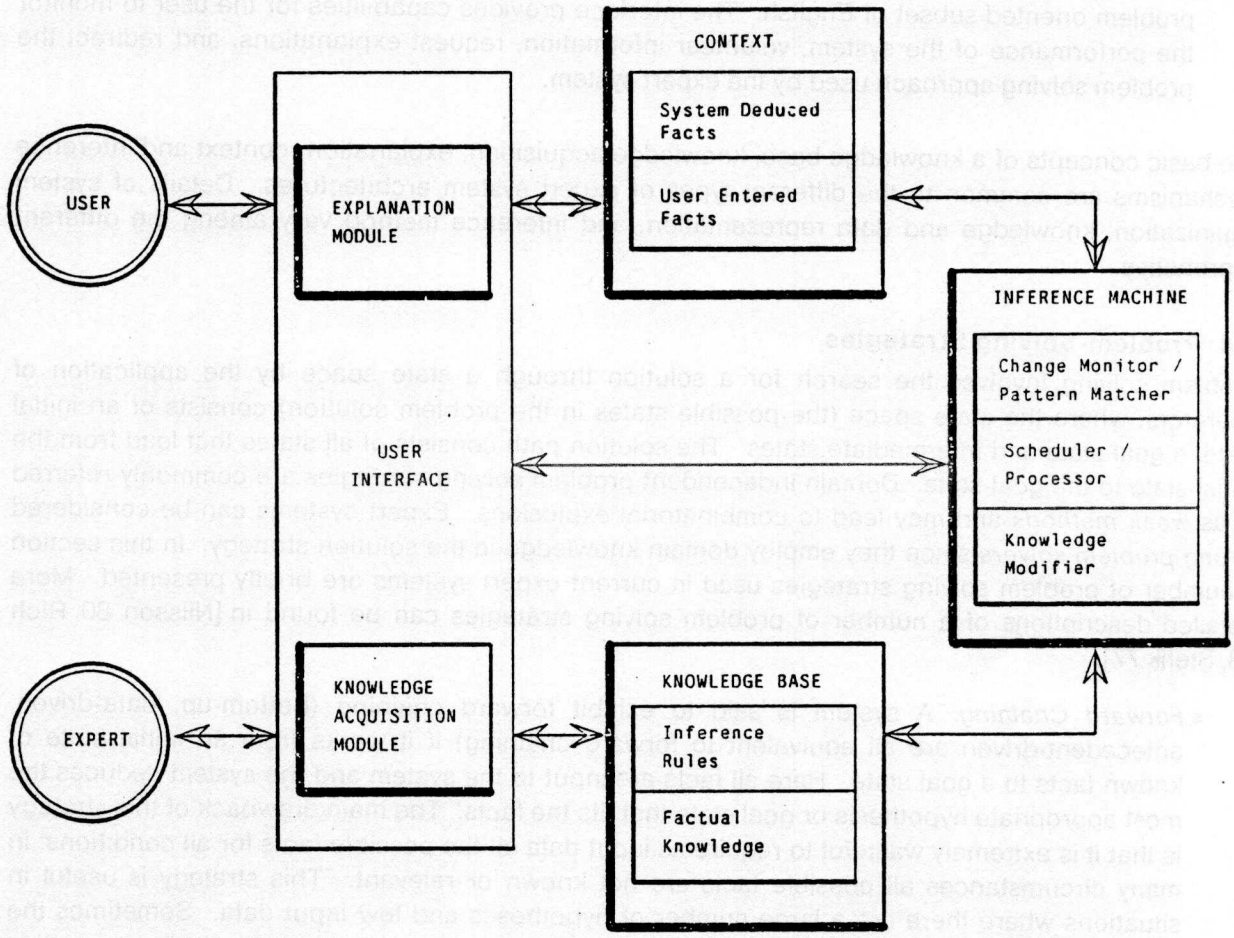

Figure 1: Schematic View of an Expert System

process continues until the problem is solved and the context is transformed into the desired goal state, or when there are no more rules remaining to be fired.

Many expert system inference machines can deal with imprecise or incomplete knowledge. Associated with all data are certainty measures indicating a level of confidence in the data. Rules conditionally fire based on the certainty of the premise. The inference mechanism can then propagate certainty about the inferences along with results of the inferences.

- *Explanation Module.* The explanation module provides the expert system with the capability to explain its reasoning and problem solving strategy to the user. At any point the user may interrupt the system and inquire what it is doing and why it is pursuing the current line of reasoning. In addition, the program can explain, in an a posteriori fashion, how any fact was deduced and how knowledge was applied.

- *Knowledge Acquisition Module.* The information in the knowledge base is in a rigid format, and the translation of knowledge obtained from experts to the required internal format may be tedious. The knowledge acquisition module aids in this task. Although it is desired that the human expert be able to directly enter knowledge into the system, this goal is currently not achieved.

- *User Interface.* The user accesses the system through a friendly interface, often using a problem oriented subset of English. The interface provides capabilities for the user to monitor the performance of the system, volunteer information, request explanations, and redirect the problem solving approach used by the expert system.

The basic concepts of a knowledge base, knowledge acquisition, explanation, context and inference mechanisms are common to the different types of expert system architectures. Details of system organization, knowledge and data representation, and inference method vary among the different approaches.

3.4. Problem-Solving Strategies

Problem solving involves the search for a solution through a state space by the application of operators, where the state space (the possible states in the problem solution) consists of an initial state, a goal state and intermediate states. The solution path consists of all states that lead from the initial state to the goal state. Domain independent problem solving strategies are commonly referred to as *weak methods* and may lead to combinatorial explosions. Expert systems can be considered *strong problem solvers* since they employ domain knowledge in the solution strategy. In this section a number of problem solving strategies used in current expert systems are briefly presented. More detailed descriptions of a number of problem solving strategies can be found in [Nilsson 80, Rich 83, Stefik 77].

- *Forward Chaining.* A system is said to exhibit forward chaining (bottom-up, data-driven, antecedent-driven are all equivalent to forward chaining) if it works from an initial state of known facts to a goal state. Here all facts are input to the system and the system deduces the most appropriate hypothesis or goal state that fits the facts. The main drawback of this strategy is that it is extremely wasteful to require as input data all the possible facts for all conditions; in many circumstances all possible facts are not known or relevant. This strategy is useful in situations where there are a large number of hypotheses and few input data. Sometimes the problem solving mechanism is guided by the events occurring during the solution process; this type of forward chaining is called *event-driven*.

- *Backward Chaining.* A system is said to exhibit backward chaining (also referred to as consequent-driven, top-down, goal-driven and hypothesis-driven) if it tries to support a goal state or hypothesis by checking known facts in the context. If the facts in the context do not support the hypothesis, then the preconditions that are needed for the hypothesis are set up as sub-goals. Essentially, the process can be viewed as a search in the state space going from the goal state to the initial state by the application of inverse operators and involves a *depth first search*.

- *Means-ends Analysis.* In means-ends analysis the difference between the current state and the goal state is determined and used to find an operator most relevant to reducing this difference. If the operator is not directly applicable to the current situation then the problem state is changed by setting up subgoals so that the operator can be applied. After an operator has been applied, the current state corresponds to a modified state. Means-ends analysis utilizes both the forward and backward chaining techniques. However, this strategy is only applicable to those tasks where the measures of differences between the various states and the operators to reduce these differences can be formulated a priori.

- *Problem Reduction.* Problem reduction involves factoring problems into smaller subproblems. The problem is represented by means of an AND-OR graph. An AND node consists of arcs

pointing to a number of successor nodes, all of which must be solved for the AND node to be *true*. For the OR node, it is sufficient for one of the successor nodes be solved; an OR node indicates that a number of alternate solutions exist for the problem. In many cases, backward chaining is used to solve the AND-OR graph. A detailed description of an algorithm (AO^*) for finding solutions in an AND-OR graph is given in [Rich 83]. This technique is very useful in tackling large complex problems.

- *Plan-Generate-Test.* The generate-and-test strategy in its purest form generates all possible solutions in the search space and tests each solution until it finds a solution that satisfies the goal condition. The plan-generate-test sequence restricts the number of possible solutions generated by an early pruning of inconsistent solutions. The pruning is achieved by the planning stage, where the data is interpreted and *constraints* are evaluated; these constraints eliminate solutions that are inconsistent.

- *Backtracking.* The problem reduction approach is applicable to problems that can be subdivided into a tree of fixed subproblems. However, in a number of practical problems it may not be possible to decompose problems into a fixed set of subproblems. A number of alternate approaches may exist. In backtracking, the problem solver backs up to other nodes, at the same level as the starting node, if no solution is found along the current path. Backtracking is incorporated in many AI languages, such as *PROLOG* [Clocksin 81]. Backtracking, in its pure form, poses a number of difficulties. To provide an efficient way of backtracking from wrong guesses, Stallman and Sussman [Stallman 77] developed the concept of *dependency-directed backtracking (DDB)*. In DDB, a record of all deduced facts, their antecedent facts along with their support justifications and the relevant rules are maintained; these records are known as *dependency records*. Support justifications are justifications for any assumptions made during the search. When the problem solver comes to a dead end, it retrieves the antecedents of the *contradiction*. Those facts which give rise to the contradiction are removed from further consideration. This strategy involves a lot of book-keeping. However, this additional book-keeping helps in a number of ways. For example, explanation of the program behavior can be extracted from the dependency records. This concept was further extended by Doyle [Doyle 78] for systems that incorporate *nonmonotic reasoning*.

- *Hierarchical Planning & Least Commitment Principle.* The concept of hierarchical planning involves developing a plan at successive levels of abstraction. For example, in design of complex systems the design space is divided into a set of levels, where the higher levels are abstractions of details at lower levels; the problem is hierarchically decomposed into loosely coupled subsystems. A number of solutions may exist for each subsystem. However, enough information may not be available to ascertain various variables of the subsystem. Further, the solution to one subproblem may depend on the decisions (or variable bindings) made in the solution of another subsystem. To minimize this dependency, it is important to defer binding decisions as far as possible. This principle is called the *least commitment principle* because variables are not instantiated (decisions are deferred) until more information about the problem space is available.

- *Constraint Handling.* If the subgoals in hierarchical planning do not interact with each other, they can be solved independently. However, in practice these subgoals do interact. The interaction between subgoals can be handled by *constraint satisfaction* methods. Constraint satisfaction methods involve the determination of problem states that satisfy a given set of constraints. Essentially, constraint satisfaction methods utilize constraints to determine the values of parameters in a *completely specified* problem. Stefik [Stefik 80] proposed an exten-

sion to the classical constraint satisfaction method (see [Mackworth 77] for a review of constraint satisfaction methods) by integrating it into hierarchical planning. This method, known as *constraint posting*, involves three stages.

1. *Constraint formulation* is the operation of adding new constraints representing restrictions on variable bindings. The constraints contain increasing detail as design progresses.

2. *Constraint propagation* is the creation of new constraints from formulated constraints. This operation handles interactions between subproblems through the reformulation of constraints from different subproblems.

3. *Constraint satisfaction* is the operation of finding values for variables so that the constraints on these variables are satisfied.

- *Agenda Control.* When a human problem solver is required to perform a number of tasks at one time, he gives a priority rating to these tasks. The task with the highest priority rating is performed first. In other words, he prepares an *agenda* of tasks. A list of justifications and a priority rating can be associated with each task. This type of control can be used for complex tasks that require focusing attention on certain parts of the problem. Agendas can also be used in systems that require several independent sources of expertise to communicate with each other.

4. Development of Expert System Applications

4.1. Scope of Expert System Applications

Expert system applications are appearing in many disciplines. However, not all tasks are amenable to expert system formulation. The following is a partial list of criteria for the evaluation of promising potential applications:

- There are recognized experts in the field whose performance is better than that of novices.

- The factual component of domain knowledge is routinely taught to neophytes who become experts by developing their own rules and empirical associations.

- Typical tasks are performed by an expert in a few minutes to several hours.

- Tasks are primarily cognitive, requiring reasoning at multiple levels of abstraction.

- Algorithmic solutions are either impractical or result in overly constrained or specialized programs.

- There are substantial benefits in applying the expert knowledge to each occurrence of the task.

A resulting system will have several desirable characteristics:

- *Usefulness.* The expert system must be capable of performing useful functions. Usefulness depends on the domain and task for which the expert system is developed.

- *Performance.* The expert system must have a high level of performance, reliability and accuracy

over a range of application cases. This requires that the program have the specialized knowledge that separates human experts from novices.

- *Transparency.* A program is transparent if it can be understood by people using the program. To have this characteristic, the expert system must be able to explain its actions and reasoning to the user.

4.2. Range of Expert System Applications

The range of potential expert system applications covers a spectrum from *derivation* or *interpretative* problems to *formative* or *generative* problems [Amarel 78]. In derivation problems, the problem conditions and description are given as part of a solution description (*goal*). The expert system completes the solution description by applying the available knowledge and rules such that the initial data and conditions are well integrated in the solution. As an example, in a derivation problem such as theorem proving, a solution hypothesis is formulated which the expert system attempts to prove by applying rules to the known data. Repeated application of rules transforms the problem statement (what to prove) to the solution state. In formation problems, conditions (*constraints*) are given in the form of properties that the solution as a whole must satisfy. Candidate solutions are generated and tested against the specified constraints. Two subclasses exist: *constraint satisfaction* in which the solution need only satisfy governing constraints, and *optimization* where an attempt is made to find the optimal solution. The design of a plan, object, or system fits this paradigm. Most actual problems are not pure formative or derivation problems, but lie somewhere between and require both techniques be used in problem solving.

The following list of problem types covers the spectrum of expert system applications. Interpretation, prediction, monitoring and diagnosis all lie at the derivation end of the spectrum while design, planning, and control lie at the formation end of the spectrum.

- *Interpretation.* An interpretation system takes observed data and explains its meaning by inferring the problem state which corresponds to the observed data. Examples are *Dipmeter Advisor*, a system for interpreting geophysical oil well log data [Davis 81, Smith 83], and *Prospector*, a system for identifying geological ore-bearing formations [Duda 79].

- *Prediction.* Starting with given situations, a prediction system infers likely consequences.

- *Diagnosis and Debugging.* Diagnosis systems infer malfunctions or system state from observed irregularities and interpretation of data. *MYCIN*, an infections disease diagnostician [Shortliffe 76], and several other medical diagnosis programs fall into this category.

- *Monitoring.* A monitor observes system behavior and compares the observations to the planned behavior to determine flaws in the plan or potential malfunctions of the system. An example is *Ventilation Manager*, a program for monitoring a patient's ventilation therapy [Fagan 79].

- *Design.* Design is the process of developing a configuration for an object which satisfies all applicable constraints. *R1* is an example of a design system which is used to configure VAX computers [McDermott 80].

- *Planning.* Planning is a design process that yields a set a actions intended to produce a desired outcome. An example is *MOLGEN*, an expert system for planning experiments in molecular genetics [Stefik 81].

- *Repair.* Repair systems plan remedies for malfunctions found through diagnosis and debugging.

- *Control.* A control system encompasses many of the characteristics of the other types of applications described above. It must interpret data, predict outcomes, formulate plans, execute the plans, and monitor their execution.

4.3. Languages and Tools for Building Expert Systems

A number of languages and tools are currently available for building expert systems. These tools can be grouped into three categories [Hayes-Roth 83].

- *General Purpose Programming Languages.* AI projects are usually implemented in a high-level language. These high-level languages need some novel features, such as facilities for experimentation with large chunks of knowledge, tentative modifications, planning and reasoning strategies. In addition, these languages need powerful abstraction mechanisms with which other higher level constructs can be built so as to make programming flexible and easy. Current expert system frameworks have been built using a number of languages, of which *LISP* [Winston 81] and *PROLOG* [Clocksin 81] are very popular among AI researchers. Bobrow [Bobrow 74] discusses some of the languages used in AI research.

- *General Purpose Representation Languages.* General purpose representation languages are programming languages developed specifically for knowledge engineering. These languages are not restricted to implementing any particular control strategy, but facilitate the implementation of a wide range of problems encompassing the derivation-formation spectrum. Some general purpose languages are: *SRL* [Wright 83], *RLL* [Greiner 80], *KEE* [Intelligenitics 84], *OPS5* [Forgy 81], *ROSIE* [Fain 81], *LOOPS* [Bobrow 83] and *AGE* [Nii 79].

- *Domain Independent Expert System Frameworks.* A domain independent expert system framework provides the system builder with an inference mechanism, from which a number of applications can be built by adding domain specific knowledge. Such systems also provide knowledge-acquisition and explanation modules to simplify the construction of the expert systems. These frameworks normally have evolved out of domain specific KBES. Hence, their control strategies are restricted to those provided in the original system. Systems under this category include: *EMYCIN* [vanMelle 79], *KAS* [Reboh 81], *HEARSAY-III* [Balzer 80], *EXPERT* [Weiss 79], and *KMS* [Reggia 82].

A number of widely used languages and tools are listed in Table 1. A detailed description of these tools is beyond the scope of this paper and the reader is referred to Part V of [Hayes-Roth 83].

4.4. Building Expert System Applications

The process of building a complete expert system consists of two distinct steps. The first step is the selection of the appropriate language or framework (inference engine, knowledge base and context structure) from among those listed in the previous section. The second step is denoted *knowledge engineering* and consists of gathering expert knowledge and augmenting a domain independent expert system framework with domain dependent knowledge to provide a fully functioning system.

Knowledge engineering is an incremental cooperative process commonly performed by two people. The first is the *domain expert* who possesses the problem solving knowledge for the problem area being addressed. The second is the *knowledge engineer* who gathers expertise from the domain expert and translates this knowledge into the format required by the expert system. A knowledge

Tool or Langauge	Developer	Knowledge Representation	Implementation Language
OPS5	Carnegie-Mellon University	Rules	*LISP* and *BLISS*
EMYCIN	Stanford University	Rules	*INTERLISP*
HEARSAY-III	USC-ISI	Rules	*INTERLISP*
EXPERT	Rutgers University	Rules	*FORTRAN*
ROSIE	Rand Corporation	Rules	*INTERLISP*
KS300	Tecknowledge Inc.	Rules	*INTERLISP*
AGE	Stanford University	Rules	*INTERLISP*
KAS	SRI International	Rules and Sematic Nets	*INTERLISP*
KMS	University of Maryland	Rules and Frames	*LISP*
KEE	IntelliGentics Inc.	Rules and Frames	*INTERLISP*
RLL	Stanford University	Frames	*INTERLISP*
PSRL	Carnegie-Mellon University	Rules and Frames	*FRANZLISP*
LOOPS	Xerox PARC	Rules and Frames	*INTERLISP-D*
KL-ONE	Rand Corportation	Semantic Nets	*INTERLISP*
C-PROLOG	Univeristy of Edinburgh	Logic	*C*

Table 1: Languages and Tools for Building Expert Systems

engineer who is also literate in the application domain is desired, as he can understand the issues involved and the nomenclature used by the domain expert [Dym 84, Fenves 84a].

The knowledge engineering process of building an expert system application is outlined below [Reboh 81, Hayes-Roth 83]. This process is similar in nature to building an algorithmic program or producing the design of an object.

- *Problem Identification.* The first step in building an expert system is to identify the problem to be solved and the characteristics of the solution. Identification of resources, domain experts, and computer facilities must be made and overall goals for the project must be set.

- *System Design.* The overall structure of the system must be selected. Based on processes used by the expert, available data, strategies, information flow, etc., a preliminary model of problem solving to be used by the system is developed. From this, a candidate domain independent expert system framework is selected and key concepts are formalized. If the selected system appears to have the correct formalisms for problem solving and knowledge representation, a more detailed analysis is undertaken to produce a detailed design of the system.

- *Knowledge Acquisition.* Knowledge acquisition is the process of gathering the expert knowledge from the domain expert. This process may be difficult. In some instances the cognitive portions of the problem solving process used by the expert have never been verbalized and are difficult to extract (the expert is not conscious of how he solves problems). In other cases the expert may feel threatened by a computerized replacement and will be reluctant to cooperate.

- *Implementation.* The knowledge engineer's tasks is then implementation; the expert's knowledge is encoded in the format required by the expert system framework. This yields an operational program.

- *Testing.* Once a prototype or partial system is developed, it is tested. The domain expert and the program are given the same problem and their problem solving behavior is compared. Flaws in the system are detected and corrected by adding or modifying the knowledge used by the program.

The process is not sequential, and a number of repetitions in the last three steps may be necessary to correct and tune the system's behavior. Adding depth of knowledge, breadth of capabilities, and improved interfaces and explanations to the prototype yields the final version of the system.

5. Potential Applications in Construction Robotics

5.1. Overview

A construction robot has to perform three functions:

1. *Sense* both the external environment and its internal state;

2. *Plan* an action based on the sensed state; and

3. *Act* according to the plan.

These three functions are supervised, coordinated and scheduled by an overall *control* function. These functions are integral to all construction robots, whether they be fixed in position (e.g., an automated concrete batch plant or prefabrication plant) or mobile (e.g., a tunnel boring machine [TBM]). Most of the processing or "reasoning" takes place in the functions of sensing, planning and control. The first generations of construction robots will undoubtedly implement these functions by pre-programmed, algorithmic means.

The Japanese construction industry uses a modified classification scheme for construction robots based on the type of planning or control:

M1	Manual Control
M2-A	Fixed Sequence
M2-B	Variable Sequence
M3-A	Playback (Teach) Control
M3-B	NC Control
M4	Intelligent Robots

Intelligent robots capable of autonomous decision-making in planning and overall control are not foreseen by the Japanese until level M4, expected to be developed in the 1990's and deployed after the year 2000.

Nevertheless, it is instructive to look ahead and attempt to identify expert system applications for augmenting or replacing some of the algorithmic techniques for some robotic functions. In the remainder of this section, some potential expert system applications that may be integral components of construction robots and relevant prototypes are presented.

5.2. Interpretation

A robot must transform its raw sensor data in a variety of ways to properly interpret them, i.e., build a reliable and usable *world model* of its environment. Many of the "low level" transformations, such as signal conditioning, smoothing, imaging, feature extraction, etc., can be performed algorithmically -- in fact, dedicated hardware or firmware is available for several of these steps. However, "high-level" transformations which extract *meaning* from the transformed sensor data are more likely to be implemented as expert systems.

In the robotic excavator (*REX*) under development at CMU for the hazardous task of excavating around leaky gas lines, it is anticipated that an array of sensors will be used. The data from the various sensing modes will be separately transformed to some common, high level representation (e.g., bounding planes of features "seen" by each sensor). It is anticipated that an expert system will be needed to integrate the separate images to provide a composite world model. The expert system will have to deal with issues such as:

- discrepancy between expected position taken from utility maps and sensed position -- an obvious heuristic is:

  IF **sensed diameter, material, etc. agree with expected**
  THEN **accept sensed position;**

- discrepancy or conflict between sensing modes;

- discrimination between valves, tees and other attachments;

- etc.

5.3. Diagnosis and Monitoring

A robot, in addition to sensing and interpreting the external environment, must also keep track of its internal condition by diagnosing and monitoring its own operating state. Here again, algorithmic control functions may be augmented by diagnostic and monitoring expert systems. Such systems can simulate the monitoring functions of an expert operator, who often takes automatic action based on the *sound* of the equipment or the *feel* of the controls. Expert systems can also make "connections" between spatially and temporarily dispersed signals and events, e.g., to distinguish between transient and hard failures, and to revise their expectations based on previous history. A critical aspect of diagnosis and monitoring will be to check the status of the robot's sensors themselves.

A prototype equipment diagnostician is *MOVER* [Fenves 83]. *MOVER* is designed to assist in the diagnosis and operation of an automated transit system. *MOVER* takes information on system failures reported to the central control computer and combines these with requested status information to determine the fault and to suggest remedial procedures to maintain operations. A second version of *MOVER* has been implemented as a prototype real-time monitor [Fenves 84b], using an event-driven strategy (see Section 3.4) to respond to changes in the sensed information.

5.4. Planning and Control

Some of the first pieces of robotic construction equipment are being designed to operate in an automated mode in hazardous environments such as mining and excavation. As outlined above, expert systems have a role in the automated interpretation and sensing of the environment of the machine. This environmental information is needed to plan the operational strategies of the machine so it can realize its goal (excavating a pipe, digging a tunnel, bolting a mine roof, etc.). An expert system can augment and eventually replace the operator to provide such control strategies.

Such an application has been proposed for the strategic and tactical control of an automated TBM [Sriram 83]. *MOVER*, described above, also is an example of a prototype operational planning system. Once a fault has been determined, it makes suggestions to the operator to facilitate the timely restoration of service. Fully autonomous construction robots, such as *REX*, will require such operational planning expert systems.

In robotic construction equipment, the complete control system will require the integration of several types of expert systems outlined above. As an example, an automated TBM requires an expert system to interpret the sensor readings to infer the tunnel geologic conditions. Diagnosis of machine faults is also made from machine sensor data. An additional expert system is required to monitor the machine performance. This data is then used to drive a planning task to select a mining strategy and detailed machine operation tactics. Such a combined control system is illustrated in Figure 2 [Sriram 83].

6. Potential Applications in Support of Construction Robots

6.1. Overview

The preceding section dealt with expert system components which may become integral parts of construction robots. In this section, the potential applications are extended to those which can serve in a support role to robotic construction.

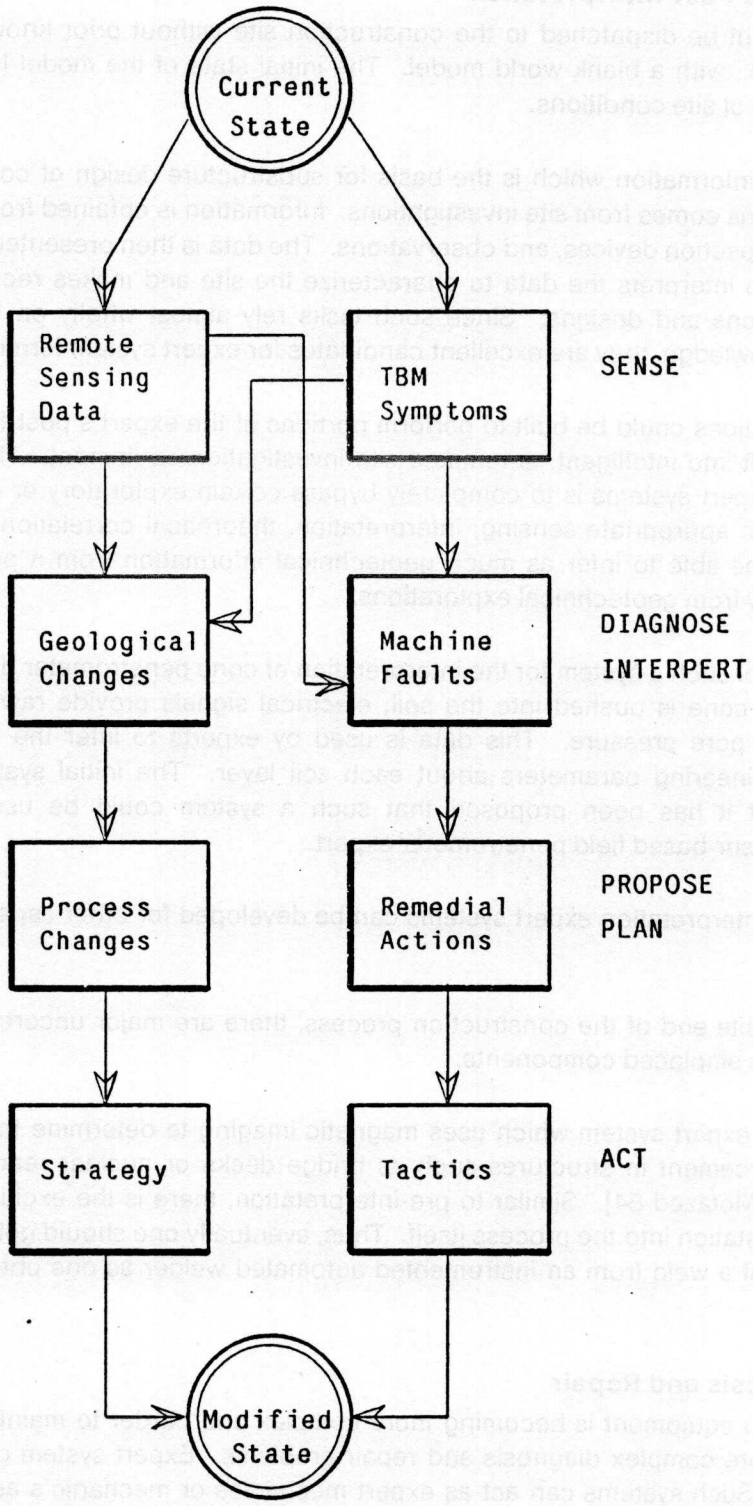

Figure 2: Automated TBM Control Process

6.2. Pre- and Post-Interpretation

A robot cannot be dispatched to the construction site without prior knowledge of the external environment, i.e., with a blank world model. The initial state of the model has to be built up from the interpretation of site conditions.

Much of the information which is the basis for substructure design of constructed facilities and for field operations comes from site investigations. Information is obtained from field and lab tests, visual and other inspection devices, and observations. The data is then presented to an expert geotechnical engineer who interprets the data to characterize the site and makes recommendations concerning possible actions and designs. Since such tasks rely almost wholly on human experts using their empirical knowledge, they are excellent candidates for expert system formulation.

Such applications could be built to perform portions of the expert's post-facto data interpretation, or could be built into intelligent, automated site investigation equipment. The most exciting possibility offered by expert systems is to completely bypass certain exploratory or evaluation operations. For example, with appropriate sensing, interpretation, theoretical correlation and empirical knowledge, one should be able to infer as much geotechnical information from a production rock-drill as one obtains today from geotechnical explorations.

An example of such a system for the interpretation of cone penetrometer data is being built [Mullarkey 83]. As the cone is pushed into the soil, electrical signals provide raw data about tip resistance, friction, and pore pressure. This data is used by experts to infer the soil stratigraphy and more detailed engineering parameters about each soil layer. The initial system will provide post-facto analysis, but it has been proposed that such a system could be used to build an automated microprocessor-based field penetrometer expert.

Similar pre-interpretation expert systems can be developed for other aspects of the external environment.

At the opposite end of the construction process, there are major uncertainties in the condition and quality of the emplaced components.

A prototype expert system which uses magnetic imaging to determine the location and size of concrete reinforcement in structures such as bridge decks or nuclear reactor containment vessels is being built [Motazed 84]. Similar to pre-interpretation, there is the exciting possibility of integrating post-interpretation into the process itself. Thus, eventually one should get as much information about the quality of a weld from an instrumented automated welder as one obtains from a post-facto weld inspection.

6.3. Diagnosis and Repair

Construction equipment is becoming more complex and harder to maintain. Automated equipment presents more complex diagnosis and repair problems. Expert system diagnosticians are a distinct possibility. Such systems can act as expert mechanics or mechanic's assistants in suggesting possible fault causes and repair and preventive maintenance procedures based on operational behavior and failure symptoms.

6.4. Planning and Design

The planning of robotic construction processes and the design of robotic equipment is itself an area for expert system applications. Problem-solving strategies such as hierarchical planning and problem reduction, discussed in Section 3.4, are directly applicable to the complex task of planning construction processes involving robotic equipment.

A significant component of planning will be the design of robotic equipment itself. Construction tasks to which robotics can be applied exhibit a large variation in the types of sensors, actuators, locomotion and control strategy that may be incorporated [Warszawski 84]. Therefore, construction robots have to be configured from a variety of component types for specific tasks. Thus, the Shimizu Construction Company fireproofing spray robot [Ueno 83] combines several levels of control from the classification given in Section 5.1:

- it executes overall operations in a fixed sequence (level M2-A)
- it moves from bay to bay under numeric control (level M3-B)
- it adjusts its position within a bay using feedback from a position sensor (level M4)
- it selects the spray pattern in variable sequence (level M2-B)
- it performs the actual spraying by playback from a manually "taught" sequence (level M3-A).

Other construction robots will undoubtedly exhibit similar complexity, so that their proper configuration and design will be a very challenging task.

Since there are very few experts in this area, an extremely useful and practical expert system would be one which could recommend appropriate combinations of robotic components based on a description of the environment, tasks and constraints.

7. Potentials in Long-Range Integration

The first generation of construction robots can be expected to duplicate the functions of present construction equipment, but with notable extension of the range of the workspace and considerable increase in safety and efficiency. The constructed facilities built with these robots will not look significantly different than those built presently, nor will their design process be significantly affected. Eventually, however, the existence of construction robotics will provide major influence on the form and functions of the facilities, and will alter the design process itself in significant ways. In other industries, this feedback is already evident; "design for manufacturability" has become a major concern in manufacturing industries, reflecting the clear need to explicitly address manufacturing concerns and constraints at the earliest, most abstract levels of conceptual design. The same trend can be expected in civil engineering design, as the means of manufacturing, i.e., constructing, facilities change drastically [Warszawski 84].

It is too early to speculate on the specific forms that this feedback from robotic construction to civil engineering may take. What is clear, however, that the feedback will be largely heuristic: certain forms, techniques, materials, etc., will be shown by experience to be more appropriate, efficient, economic, etc., for robotic construction than others. Expert systems are the most appropriate computer aids developed so far for representing and processing such heuristic knowledge. It is to be expected, therefore, that computer-aided civil engineering will increasingly use knowledge based systems to integrate knowledge about robotic construction into the design process.

8. Future Directions

Work in producing expert systems for engineering applications is progressing in parallel in two areas.

- *Frameworks and Basic Capabilities.* Research is underway in developing better expert system frameworks. Such frameworks provide better knowledge representation schemes, alternative inference techniques, and alternative mechanisms for dealing with uncertain or partial data. Extensions to provide interfaces to algorithmic programs, database management systems, graphical displays, and connections to sensors are also being developed. Better user interfaces and knowledge acquisition modules will improve the usability of expert systems and will reduce application development cost and time.

- *Applications in New Domains.* Numerous prototype expert system applications are under development in many civil engineering domains. This work leads to demands for better expert system frameworks and results in better techniques for applying expert system technology. This work also leads to a more fundamental understanding of the decision making process in the application domains. More importantly, the prototype systems are moving out of the laboratory and into practice.

Continued work will improve the technology and applications of knowledge based expert systems.

9. Summary

Knowledge based expert systems were introduced and presented as an alternative to traditional algorithmic programs. The overall structure, use, and operation of expert systems was presented.

By capturing the knowledge used by experts in problem solving, expert systems are capable of solving problems which, to date, have avoided computerization. There is a wide variety of potential applications in civil engineering. The emergence of construction robots will create the need for many further applications. Because of the wide range of robotic tasks and environments in construction, expert systems will be needed to create "intelligent" robots, as well as to provide feedback from robotic construction to design. Additional research and development of expert systems will demonstrate how this powerful new computer based technology can aid in the civil engineering and construction robotics domains.

10. References

[Amarel 78] Amarel, S., "Basic Themes and Problems in Current AI Research," *Proceedings, Fourth Annual AIM Workshop*, Ceilsielske, V.B., Ed., Rutgers University, pp. 28-46, June 1978.

[Balzer 80] Balzer, R, Erman, L.D. London, P., and Williams, C., "Hearsay-III: A Domain Independent Framework for Expert Systems," *Proceedings, First Annual National Conference on Artificial Intelligence*, pp. 108-110, 1980.

[Bobrow 74] Bobrow, D.G., and Raphael, B., "New Programming Languages for AI Research," *Computing Surveys,* Association for Computing Machinery (ACM), Vol. 6, No. 3, pp. 153-174, 1974.

[Bobrow 83] Bobrow, D., and Stefik, M., *The Loops Manual,* Xerox PARC, 1983.

[Buchanan 69] Buchanan, B., Sutherland, G., and Feigenbaum, E.A., "Heuristic DENDRAL: A Program for Generating Explanatory Hypothesis in Organic Chemistry," in *Machine Intelligence,* American Elsevier, New York, 1969.

[Clocksin 81] Clocksin, W.F., and Mellish, C.S., *Programming in Prolog,* Springer-Verlag, New York, 1981.

[Davis 81] Davis, R., et al., "The Dipmeter Advisor: Interpretation of Geologic Signals," *Proceedings, Seventh International Joint Conference on Artificial Intelligence*, pp. 846-849, 1981.

[Doyle 78] Doyle, J., *Truth Maintenance Systems for Problem Solving,* unpublished Master's Thesis, MIT, Cambridge, MA, 1978.

[Duda 79] Duda, R.O., et al., *A Computer-Based Consultant for Mineral Exploration*, Final Report SRI Project 6415, SRI International, 1979.

[Duda 83] Duda, R.O., and Shortliffe, E.H., "Expert Systems Research," *Science,* Vol. 220, pp. 261-268, April 1983.

[Dym 84] Dym, C.L., *New Approaches to Computer-Aided Engineering*, Technical Report ISL-84-2, Xerox Palo Alto Research Center (PARC), April 1984.

[Erman 80] Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R., "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty," *Computing Surveys,* Association for Computing Machinery (ACM), Vol. 12, No. 2, pp. 213-253, February 1980.

[Fagan 79] Fagan, L.M., et al., "Representation of Dynamic Clinical Knowledge: Measurement Interpretation In the Intensive Care Unit," *Proceedings, Sixth International Joint Conference on Artificial Intelligence*, Tokyo, pp. 260-262, 1979.

[Fain 81] Fain, J., et al., *The ROSIE Reference Manual*, Technical Report N1-1647-ARPA, Rand Corporation, Santa Monica, CA, 1981.

[Fenves 83] Fenves, S.J., Bielak, J., Rehak, D.R., Rychener, M., Sriram, D., and Maher, M.L., *Feasibility Study of an Expert System for Peoplemover Operation & Maintenance*, Technical Report, Civil Engineering and Construction Robotics Laboratory, Department of Civil Engineering and Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, January 1983.

[Fenves 84a] Fenves, S.J., *Who Will Write Expert Systems: A Historical Perspective*, unpublished , 1984.

[Fenves 84b] Fenves, S.J., Maher, M.L., Rehak, D.R., Rychener, M.D., and Sriram, D., *PDS MOVER: Development of a PDS Expert System Prototype for Peoplemover Operation & Maintenance*, Technical Report, Civil Engineering and Construction Robotics Laboratory, Department of Civil Engineering and Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, May 1984.

[Forgy 81] Forgy, C.L., *OPS5 User's Manual*, Technical Report CMU-CS-81-135, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, July 1981.

[Fox 82] Fox, M.S., *SRL: Schema Representation Language*, Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, 1982.

[Greiner 80] Greiner, R., and Lenat, D.B., "A Representation Language Language," *Proceedings, First Annual National Conference on Artificial Intelligence*, pp. 165-168, 1980.

[Hayes-Roth 83] Hayes-Roth, F, Waterman, D., and Lenat, D., *Building Expert Systems,* Addison-Wesley, Reading, MA, 1983.

[Intelligenitics 84] *KEE: Knowledge Engineering Environment,* IntelliGentics, Inc., 1984.

[Mackworth 77] Mackworth, A.K., "Consistency in Networks of Relations," *Artificial Intelligence,* Vol. 8, pp. 99-118, 1977.

[McDermott 80] McDermott, J., *R1: A Rule-Based Configurer of Computer Systems*, Technical Report CMU-CS-80-119, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1980.

[Moses 71] Moses, J., "Symbolic Integration: The Stormy Decade," *Communications of the Association for Computing Machinery (CACM),* Association for Computing Machinery (ACM), Vol. 14, No. 8, pp. 548-560, August 1971.

[Motazed 84] Motazed, B., *Expert Imaging of Magnetic Objects*, unpublished Ph.D. Thesis Proposal, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, January 1984.

[Mullarkey 83] Mullarkey, P.W., *Development of an Expert System for Interpretation of Geotechnical Characterization Data from Cone Penetrometers*, unpublished Ph.D. Thesis Proposal, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, November 1983.

[Nau 83] Nau, D.S., "Expert Computer Systems," *Computer,* IEEE Computer Society, Vol. 16, pp. 63-85, February 1983.

[Nii 79] Nii, H.P., and Aiello, N., "AGE (Attempt to Generalize): A Knowledge-Based Program for Building Knowledge-Based Programs," *Proceedings, Sixth International Joint Conference on Artificial Intelligence*, pp. 645-655, 1979.

[Nii 82] Nii, P., Fiegenbaum, E., Anton, J., and Rockmore, A., "Signal-to-Symbol Transformation: HASP/SIAP Case Study," *AI Magazine,* Vol. 3, No. 2, pp. 23-35, Spring 1982.

[Nilsson 80] Nillson, N.J., *Principles of Artificial Intelligence,* Tioga Publishing Company, Palo Alto, California, 1980.

[Reboh 81] Reboh, R., *Knowledge Engineering Techniques and Tools in the Prospector Environment*, Technical Report 8172, SRI International, June 1981.

[Reggia 82] Reggia, J.A., and Perricone, B.T., *KMS Manual,* Department of Mathematics, University of Maryland, 1982.

[Rich 83] Rich, E., *Artificial Intelligence,* McGraw Hill, 1983.

[Shortliffe 76] Shortliffe E.H., *Computer-Based Medical Consultations: MYCIN,* American Elsevier, New York, 1976.

[Simon 81] Simon, H.A., *The Sciences of the Artificial,* Second Edition, MIT Press, Cambridge, MA, 1981.

[Smith 83] Smith, R.G., and Baker, J.D., "The Dipmeter Adivsor: A Case Study in Commercial Expert System Development," *Proceedings, Seventh International Joint Conference on Artificial Intelligence*, pp. 122-129, August 1983.

[Sriram 82] Sriram, D., Maher, M., Bielak, J., and Fenves, S., *Expert Systems for Civil Engineering - A Survey*, Technical Report R-82-137, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, July 1982.

[Sriram 83] Sriram, D., Mullarkey, P., Bielak, J., and Fenves, S.J., *Potentials for Artificial Intelligence and Robotics in Tunnel Boring Machines*, Technical Report, Civil Engineering and Construction Robotics Laboratory, Department of Civil Engineering and Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, January 1983.

[Stallman 77] Stallman, R., and Sussman, G.J., "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis," *Artificial Intelligence,* Vol. 9, pp. 135-196, 1977.

[Stefik 77] Stefik, M., and Martin, N., *A Review of Knowledge Based Problem Solving as a Basis for a Genetics Experiment Designing System*, Technical Report STAN-CS-77-596, Department of Computer Science, Stanford University, March 1977.

[Stefik 80] Stefik, M., *Planning With Constraints*, Technical Report STAN-CS-80-784, Department of Computer Science, Stanford University, January 1980.

[Stefik 81] Stefik, M., "Planning with Constraints (MOLGEN 1)," *Artificial Intelligence,* Vol. 16, pp. 111-140, 1981.

[Ueno 83] Ueno, T., and Yoshida, T., "Robotization of Spraying Work for Fireproofing," *Robot,* No. 38, 1983, [in Japanese].

[vanMelle 79] van Melle, W., "A Domain Independent Production-Rule System for Consultation Programs," *Proceedings, Sixth International Joint Conference on Artificial Intelligence*, pp. 923-925, August 1979.

[Warszawski 84] Warszawski, A., *Applications of Robotics to Building Construction*, Technical Report, Civil Engineering and Construction Robotics Laboratory, Department of Civil Engineering and Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, May 1984.

[Weiss 79] Weiss, S.M., and Kulikowski, C.A., "EXPERT: A System for Developing Consultation Models," *Proceedings, Sixth International Joint Conference on Artificial Intelligence*, pp. 942-947, 1979.

[Winston 81] Winston, P.H., and Horn, B.K.P., *LISP,* Addison-Wesley, Reading, MA, 1981.

[Wright 83] Wright, M., and Fox, M., *SRL: Schema Representation Language*, Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, December 1983.