Proceedings of the 1984
Winter Simulation Conference
S. Sheppard, U. Pooch, D. Pegden (eds.)

KEYNOTE ADDRESS

ARTIFICIAL INTELLIGENCE AND SIMULATION

Robert E. Shannon
Texas A&M University

ABSTRACT - Artificial Intelligence is the latest buzzword and one of the hottest topics in the scientific community today. Some experts are proclaiming that Artificial Intelligence (AI) has already emerged as one of the most significant technologies of this century. Proponents are declaring that it will completely revolutionize management and the way we use computers. If these claims are even half true, then AI is bound to have a profound effect upon the art and science of simulation. The purpose of this paper is to provide a current overview of this rapidly evolving field, examine the potential of AI in simulation and the inevitability of it. We propose to explore the probable impact as well as forecast the directions it is likely to take.

INTRODUCTION:

Ever since the advent of the modern digital computer, scientists have speculated about and argued over the possibility that computers could behave in a way that would be perceived as intelligent. Artificial Intelligence (AI) has its roots in the speculative essays by Turning on the powers of computers (1). AI as we know it today, is the result of a meeting convened in 1956 by ten scientists interested in symbolic computation at Dartmouth College (2). At the conclusion of that meeting, the scientists confidently predicted that in 25 years, we would all be involved in recreational activities, while computers would be doing all the work. In 1981, at the International Joint Conference on AI in Vancouver, Canada, a panel of five of these same scientists recalled that conference and their overly optimistic forecasts. However, from its early beginnings in computer laboratories 28 years ago, the AI field has matured to practical commercial applications within the last few years. Artificial Intelligence is no longer a pie-in-the-sky notion. Systems have recently come to market that can simulate some of the characteristics of human thought - the ability to learn, reason, solve problems and understand ordinary human language.

AI tools and development systems are now becoming available, and techniques are now sufficiently perfected for early applications. The importance of AI is being internationally recognized and substantial sums of money in the U.S. and abroad are being committed to research and for developing AI applications. The Japanese Fifth Generation Computer Project and the U.S. Microelectronics and Computer Cooperative here in Texas, have committed almost $1 billion in research to advance AI.
Companies (founded mostly by AI researchers) have been formed to exploit applications. Computer, electronic, oil, and other large diversified companies have set up their own AI groups. Software for micro-computer applications has already begun to appear (3). The military has jumped into the field in a big way. The U.S. Defense Science board views it as one of the technologies that has the potential for an order of magnitude improvement in mission effectiveness. Future predictions indicate that ten years from now, half the computers sold will contain artificial intelligence components as well as arithmetic components, and will be called "logic machines." In short, we are seeing the beginnings of a multi-billion dollar industry and a revolution in the way we think about and use computers.

Unless a lot of people are wrong, the technology being developed in the AI field is going to significantly affect computers, software, problem solving and management. If this is true, then it is obvious that it will also affect the art and science of simulation. It appears that profound changes are going to come as a result of artificial intelligence. As simulation professionals, we must not be intimidated, but must think about the potential of AI, the inevitability of it and be aware of what is going on so that we can exploit it in our own enterprises. In In the remainder of this paper we will try to explore what AI is and how it is likely to impact simulation.

ARTIFICIAL INTELLIGENCE (EXPERT SYSTEMS):

Artificial Intelligence systems are computer programs that simulate some of the characteristics of human thought - the ability to learn, reason, solve problems and understand ordinary human language. With the advent of relational databases, computers changed from mere data processing machines into decision support systems. Since these relational systems facilitate finding and using the relationships between the various types of data stored, they can be used to develop applications that will aid the analysis and decision making process.

AI systems break down into two basic categories: natural language query systems and knowledge-based systems. Natural language query systems allow the user to ask questions of a database without using special commands. "Expert" or knowledge-based systems are designed to compile the experience of any number of experts in a given field into a series of "if-then"rules. These rules then draw inferences and suggest to the user a course of action to deal with a given problem. The remainder of this discussion will deal with "expert" or knowledge-based systems.

Expert Systems are problem-solving computer programs that can reach a level of performance comparable to that of a human expert in some specialized problem domain. Such systems typically provide formats or languages for storing or representing both declarative knowledge (data) and procedural knowledge

(relationships) as well as control structures for manipulating this knowledge through friendly user interfaces. Thus most expert systems organize knowledge on three levels: data, knowledge base, and control.

A. Data Level - declarative knowledge is data or facts about a particular problem being solved and the current state of affairs. There are several ways to represent declarative knowledge (databases), among which are:

1. Predicate Calculus (logic): Simple declarative facts can often be represented as predicates. For example, "John gave Mary the book", can be represented as GIVE (John, Mary, book). Or given x,y,z if R(x>y) & R(y>z) then R(x>Z) is a statement that the statement that the relation R is transitive. Predicate calculus is not concerned with mathematics but with formal logic. It has been found particularly useful in AI because thru it we can recast ordinary English sentences into a formal representation that can be handled by a computer and compared to other information.

2. Frames: Frames are data structures in which all knowledge about a particular object or event is stored together. Most frame based knowledge representation schemes include the idea of having different types of frames for different types of objects, with fields or slots in each frame to contain the information relevant to that type of frame. For example, a frame for a book might be a data structure with slots for the author, title, and publication date as well as for number of pages and color of cover.

3. Semantic Networks: Semantic nets are like frames in the sense that knowledge is organized around the objects being described, but here the objects are represented by nodes in a graph and the relations among them are represented by labeled arcs. Certain types of relationships can best be made clear by a graphical presentation. For example, computer programs are often difficult or impossible to understand when written as lines of code, even in a high level language. But when presented as a flow chart, it can be more easily followed. So too, even if predipredicate calculus can be used to express the same relationships, the connections and interactions between various elements is often not clear when a long string of logic expression follows one after another.

B. Knowledge-base Level - domain specific i.e. knowledge specific to the particular kind of problem that the system is set up to solve. This knowledge is usually procedural (or relational) in the sense that it tells how the data for a problem can be manipulated in order to go about solving the problem. The power of an expert system lies in the specific knowledge of the problem domain. Most AI systems contain hundreds of rules obtained by seeking out the knowledge possessed by the experts and then linking this knowledge to form rule networks. Again, there are several ways in which this knowledge can be represented, such as:

1. Conventional computer program: If the solution procedure is well understood, deterministic and can be programmed as an algorithm, then a conventional program can be written to manipulate the data to get a solution.

2. Pattern-invoked programs: Here we encode the domain dependent knowledge in the form of operators or pattern invoked programs which are activated by the control structure when certain conditions hold in the data. Some of these are in the form of state-space search programs where we want to find the path from some initial state to some final or goal state by applying operators to transform states into other states.

3. Production Rules: These are programs of the form "IF <condition> THEN <primitive action>". The condition is usually a conjunction of predicates that test properties about the current state, and the primitive action then changes the current state.

4. Logical Representation: Sometimes the knowledge can be stored in the form of an AND/OR graph. A solution is then found by doing a tree search until a set of assertions is found that provides a solution.

5. Conditional Probabilities: Knowledge is sometimes represented in terms of conditional probabilities that various events will occur given that other events have occurred.

C. Control Level - the computer program that makes decisions about what is the question to be answered, and then how to use the data base and knowledge base to solve the problem at hand. In an ideal AI system, the control structure will interpret and understand the user's request and then determine what is needed in terms of data input, techniques to process the information and the type of information to be output. As pointed out by Young and Mayer (4), the control level will consist of three parts - a natural language processor, a resolution processor and a command sequence generator. The natural language processor will interpret for the computer the requests given by the user in normal language form. The resolution processor will decide whether the question can be answered by retrieval from a data base or will require the running of a model. If the running of a model is required it will determine which one (assuming multiple models), the experiment required to generate the correct answers, and the form of the output. The command sequence generator will retrieve the necessary information and software modules, execute the software modules in the proper sequence, analyze the models output and provide the requested answers in the appropriate form. The command sequence generator will also provide back to the resolution processor an explanation of the line of reasoning and data used to produce the results.

The Command Generator organizes and controls the steps taken to solve the problem. The steps or rules invoked are actuated by patterns such as system status, chaining of IF-THEN rules or probabilistic branches. The control strategy used is a function of the problem to be solved and several approaches used include:

1. Forward Chaining: If the solution starts from a set of data and conditions and moves towards some unknown conclusion, it is called model, data, event or antecedent driven. Almost all simulation models as we know them today, use this solution method. If the model is not time dependent, then state-space search methods such

as branch and bound, hill climbing or tree searching techniques are usually employed.

2. Backward Chaining: If the desired conclusion is already known (e.g. it is a goal to be achieved), but the path to that conclusion is not known, then working backwards is called for and the model is said to be goal or expectation driven. Dynamic programming methodology is usually employed.

3. Problem Reduction: In this technique, the problem to be solved is partitioned or decomposed into subproblems that can be solved separately. The combining of the solutions to the subproblems then sometimes provides a solution to the overall problem. This approach, which yields problems with a smaller search space, is applicable for problems in which a number of non-interacting tasks have to be done to achieve a goal. Dendral (5) is an example of a forward chaining model using problem reduction, while Mycin (6) uses backward chaining.

4. Constraint Propagation: In this problem solving technique, the set of possible solutions becomes further and further constrained by rules or operators that produce "local constraints" on what small pieces of the solution must look like. Constraints are viewed as relationships (or subgoals) that must be satisfied. For a discussion and examples see references (7) and (8).

WHAT'S DIFFERENT:

The reader no doubt has noticed that simulation languages and simulation models contain many of the ideas being used in AI. For example, the ability: of entities or processes to carry with them (and change dynamically) attributes which describe their characteristics (FRAMES); to dynamically modify flow of entities thru the system (PRODUCTION RULES AND CONDITIONAL PROBABILITIES); to change the system based upon state variables (PATTERN-INVOKED PROGRAMS) ; and to represent knowledge about the system in the form of a network. A natural question then is how would an AI simulation system vary from just a well written simulation model as we know it today? This is a difficult question to answer but we will try.

First of all, in an AI system the data base, knowledge base, and control structure are separate and each can be modified easily without affecting the others. Most simulation models today have integrated information and control, although some languages have begun to separate them into two or three parts (model, experiment and output analysis).

A second difference between simulation models as we know them today and an expert simulation system would be in certain characteristics. Some of the characteristics of current conventional simulations are:

a) Primarily numeric

b) Algorithmic (solution steps explicit)

c) Integrated information and control

d) Several steps of modeling process done separately or out side of the model (e.g. testing input data for goodness of fit; determining sample sizes; designing the the experiment to be run).

e) Model can't do anything which is not pre-planned (i.e. user must instruct the program what to do).

An expert simulation system, on the otherhand, would:

a) Have many symbolic processes

b) Use heuristic search (solution steps not implicit)

c) Command structure usually separate from knowledge domain

d) Have the expertise built into the model so that decisions by the user would be minimized.

e) The model would be able to learn from its own experience.

A third method of looking at the difference is in the way an AI simulation system would be used. Michie (9) observes that ideally there are three different user modes for an expert system:

1. improving or increasing the system's knowledge -- user as tutor

2. getting answers to problems -- user as client

3. harvesting the knowledge base for human use -- user as pupil.

In the user as tutor mode, the simulation system should provide model development capabilities that will allow users to easily develop and modify representations of real or proposed systems without having to spend 100 or more hours learning a new modeling language. There should be a separate data entry interface that allows users to enter descriptive information about the system or entities without requiring they know the form of the model or the format eventually required for the data. It is in this mode that we build the knowledge base.

In the second mode, user as client, the simulation should allow the user to pose questions in his or her native language and the system decide what kind of experiment to run, required sample sizes etc. The user (or the system) should be able to make simple modifications to the model without requiring direct programming changes in the model's computer code. The simulation system should be able to get from corporate data bases the information it needs to answer the questions (i.e. routings, processing times etc.) without being directed by the user as well as the current status of the system. In addition, the user should be able to select the types and form of output desired.

In the third, or user as pupil mode, the simulation system should be capable of learning and able to modify itself as it goes along. This will require that it be capable of backtracking or "time warps" during execution and that it be capable of either forward or backward chaining. This may require that we use multiple parallel processors. The system, for example should be capable of discovering the optimal schedule or how to reach a pecified goal. We can

already do this to a limited extent using response surface methodology in optimization experiments. However, most models today simply give us the probable results for a given scenario (inputs).

In summary, although there are similarities between what we are doing today and the goals of an expert simulation system, there are important differences. The primary one in my opinion is the desire to build into the model most of the decisions that are now made by the simulation expert. Today, in order to use simulation correctly and intelligently, the practitioner is required to have expertise in a number of different fields. This generally means separate courses in modeling, design of experiments, statistics and a simulation language. This translates to about 600 hours of instruction (1200 or more if self study) and that is only to gain the basic tools. In order to really become proficient, the practitioner must then go out and gain real world, practical experience.

WHAT NEXT:

The problem with moving from where we are today, to the AI based Expert Simulation Systems of tomorrow, is not so much in what we do not know as it is that we simply haven't started thinking in the right terms. The progress of our knowledge about the art and science of simulation during the past 10 years has been phenomenal. No place is this more evident than in the proceedings of this conference. In fact it is this very explosion of knowledge that has become one of our stumbling blocks. No one, not even the "experts" can today know everything that is relevant about simulation. We are already be ginning to specialize (language developers, random number genera tion, output analysis etc.). Furthermore, the situation is bound to get worse. The rate at which new understanding and knowledge is being gained is rapidly accelerating. The time has come to begin making that knowledge easily usable by the field practitioner and the means is thru AI based Expert Simulation Systems. The question, in my opinion, is not can it or will it be done but rather when and by whom.

You have no doubt heard something about the Japanese "Fifth Generation Computer Project" and America's partial answer in the Austin, TX based Microelectronics and Computer Cooper ative (MCC). Both of these efforts are directed towards advancing the state of the art in both hardware and software in order to implement AI. You may have wondered where the term, "Fifth Generation" comes from and what it means. In terms of software, it is concerned with the way we communicate and interact with the computer. The first generation was machine language and the second assembly languages. The third generation languages included non-procedural, higher level languages like FORTRAN, Basic, Cobal, C, Pascal and Lisp (the building blocks of today's packaged software tools).

Fourth generation languages is an umbrella term including several categories of software. There are at least three major areas: presentation languages (formal query, natural query, reporting, graphics etc.); specialty languages that focus on a specialized function (spreadsheets, modeling, analysis, simulation etc.); and application generators that deal with data capture, modification and definition to build complete applications. Although these languages are marketed as end user tools, the fact remains that they have to be learned and if you give them to a non-programmer, their going to have a very difficult time. In reality, 4th generation languages are excellent productivity tools for programmers.

The fifth generation languages will have several characteristics that will separate them from the 4th generation namely: use English commands and extremely easy to learn, use and support; print their own documentation, simplifying updates or changes; easy to transport from computer to computer; and use virtual memory based design. In addition, if non-prog rammers are going to be able to develop their own applications, then the 5th generation languages will have to completely automate the programming task. Systems which present forms to be filled in, menu's, icons, or interactive queries will be required. The desired application will then be built in response to the user's answers without further concern by the user. In other words, the fifth generation will integrate the tools developed in the 4th generation and capture the knowledge of the expert programmer as well as that of the simulation modeling expert.

It is apparent that we are already in the transition phase from 4th to 5th generation simulation systems. The increasing use of interactive graphical model construction and data input; graphical and animated output analysis; the separation of modeling, experimental and output analysis frames; the imbedding of more and more of the statistical analysis within the language; all of these are the first tentative steps and will continue. But at some point in the not too distant future, I expect to see the development of some new, radically different simulation systems. I do not think they will be merely enhancements of existing languages because of the inherent limitations imposed upon existing languages by the 4th generation languages they are based upon.

What 4th generation languages will be used to write the fifth generation simulation language? I don't really know. In the United States, LISP and it's dialects has been far and away the favorite for AI research and applications while in Europe and the Far East, PROLOG prevails. The Japanese have decided upon PROLOG for their fifth generation projects. Recently, several researchers have been pointing out the benefits of using APL and Forth. But if I were to guess right now, the edge would go to PROLOG. PROLOG is a much smaller program than LISP (although not as small as Forth) and has been implemented on a variety of computers including microcomputers. It's execution speed is faster than LISP and is efficient in execution. PROLOG has been very popular in Europe and as previously mentioned is now targeted as the language of the Japanese Fifth Generation Computer Project. PROLOG was originally developed for natural language understanding applications, but has since found use in virtually all AI application areas. It's design is well suited to parallel search and is therefore an excellent candidate for future powerful computers incorporating parallel processing. Japan has announced that it intends to build a sequential PROLOG based personal computer by 1986, featuring 10K logical inferences per second, and by 1999 a parallel processing computer running PROLOG at 1 billion inferences per second.

So in summary, what we can expect to see is the continuing enhancement and integration of existing 4th generation simulation languages to make them more

powerful and easier to use until they bump into the barriers imposed by their underlying programming language, and then the emergence of new, AI based, 5th Generation, Expert Simulation Systems.

AN EXPERT SYSTEM FOR MANUFACTURING:

It is clear that expert systems will for some time into the forseeable future be designed for use in a particular area of decision support. For example, an expert system might be designed for support of management in manufacturing decisions. It appears to the author that simulation will be used in three distinct modes in manufacturing, as follows:

1. As a design and analysis aide for factory layouts, new equipment decisions, exploring alternative operating policies, seeking explanations for problem areas etc.. This is the traditional role currently played by simulation models.

2. As a tool for scheduling, particularly with automated systems. Such a use would allow the decision maker to explore and plan changes to the existing schedule and or to find the most optimal schedule starting with current conditions. For example, current conditions might include the fact that a particular piece of equipment was broken down. The model would then generate a new alternative schedule for use until the equipment was repaired.

3. As a part of a real time, on-line control system. Such a system would periodically be activated, read the current conditions from a data base, project the schedule forward and then depending upon the results leave well enough alone, modify the schedule, or call for human intervention.

Let us hypothesis what an AI based expert simulation system might look like. With the caveat that the one thing a forcaster can be sure of is that he or she is almost surely wrong, we will describe a system for the simulation of manufacturing systems in the following scenario:

The main menu upon bootup is:

Query current conditions
Retrieve existing model
Directory of existing models
View layout of model
Create new model
Change model
Define an experiment
Retrieve an existing experiment
Directory of existing experiments
Query current schedule
Develop new schedule
Execute model
Exit Program

(A) Building The Model:

Upon selecting "Create new model"' (with the cursor) a question appears on the screen asking for the name of the new model. The second question follows, "What is the physical size of the plant area to be simulated?" The purpose of this question is to allow the program to properly scale the layout and modeling icons to be used in designing the model.

The next thing that happens is the appearance on the screen of the outline of the physical space which will contain the model upon the screen and a series of questions allowing the user to redefine the outline of said space i.e. doors, internal partitions, odd shapes etc.

After the outline of the space has been defined to the users satisfaction, we are ready to begin defining the model. A list of available icons will appear on the right hand of the screen such as:

LATHE
DRILL
MACHINING CENTER
GRINDER
SPOT WELDER
MILL
ROBOT
CONVEYOR
USER DEFINED
ETC.

Using the cursor, the user will select the component to be placed in the model. The program will then ask the user for the station number of the new component and to place the cursor at the location on the layout where the component is to be placed and hit "return". The appropriate icon will then appear at that position (properly scaled). The user will proceed to select, number and place the remaining components of the system to be modeled (with the ability to quickly check the distances between components). In the case of certain components such as cranes, conveyors etc. the program will require additional in formation such as starting and ending points (which it will automatically solicit). In other cases it will need to solicit the type of lathe, robot etc.. When the physical layout (showing the relationships of the components) is completed to the users satisfaction it will be automatically saved.

The program will then start back thru the stations soliciting the appropriate information to further define each station (i.e. size of buffer storage, operator number etc.). The questioning will depend upon the type of component being defined and will occur in the opposite (diagonal) quadrant of the screen from the location of the component. Thus, the user is seeing the location of the component and the two adjacent quadrants (three fourths of the layout) while answering the questions. At the conclusion of this phase, the model has been defined and is automatically saved.

(B) Designing The Experiment:

The design of an experiment to be run with the model is driven by the goal or purpose of the study. Therefore, the first question asks the purpose of the experiment:

EVALUATION: determining how good a proposed system design performs in an absolute sense when evaluated against specific criteria.

COMPARISON: comparing competitive systems designed to carry out a specified function, or comparing several proposed operating policies or procedures.

PREDICTION: Estimating the performance of the system under some projected set of conditions.

SENSITIVITY ANALYSIS: determining which of many

factors are most significant in affecting overall system performance.

OPTIMIZATION: determining exactly which combination of factor levels will produce the best overall response of the system.

FUNCTIONAL RELATIONS: establishing the nature of the relationships between one or more significant factors and the system's response.

TRANSIENT BEHAVIOR: looking for specified transient behavior such as bottlenecks, excessive queue buildup utilization inbalances etc..

The purpose of the study implies the statistical objectives to be achieved by the experiment. This sets the line of interactive questioning that will follow to define the experiment. One of the difficulties is the fact that a particular study might have more than one "measure of effectiveness". In such cases it will be necessary for the user to designate one as the primary and the others as secondary. The primary measure of effectiveness will be used by the program to calculate required sample sizes etc.

As stated earlier, the goal selected will determine the line of interactive inquiry that the program pursues. For example, lets assume the user says the goal is to explore functional relationships. This Implys that a factorial design will be required. The program would then solicit information as to what variables are of interest, the highest and lowest values of the range of interest, the highest and lowest values of the range if interest, whether each variable is a quanitative or qualitative etc.. From this information (as well as the measure of system effectiveness to be used), the program would figure out a factorial design to be used as well as the number of a replications to be run. the user would then be solicited for other statistics that might be of secondary interest.

This phase of the process would also have to solicit information as to what is to be considered a sample i.e. a day's, week's, month's or year's production, each batch of similar parts, etc.. From the above dialog, the program would design the experiment to be run to obtain the answers needed and automatically save it as *.exp.

(C) Specifying The Input:

It is assumed that a corporate data base has been established which contains information on each part to be manufatured in the plant. For each part, the data base would contain routing sheets (the machines required to make the part and their sequence), setup times for each machine, machining/processing times, load/unload times etc.. These times might be stored as constants, ranges, or as probability distribution (with the appropriate parameters). The user would be asked to specify which parts (by part number, family group etc.) are to be included in the simulation, batch sizes (if appropriate), arrival rates to the system and due dates. Arrival rates, batch sizes, due dates etc. could be read in from a file (the proposed schedule or a historical record), specified as random (following some specified probability distribution), or as a constant.

The program would then go to the database and pull out the necessary information on the parts to be included in the simulation, and associate the information as attributes of the part.

(D) Running the Experiment:

When we were ready to run the experiment, the Command Generator would take over and start the simulation. After it had run awhile, it would automatically look at the data being generated and decide if some of the early data needs to be truncated and how much. In the case of experiments designed to estimate system parameters (means, variances, proportion etc.), the Command Generator would automatically test the data being generated for autocorrelation, and if present, calculate the needed sample and/or batch sizes for the desired accuracy. It would then forcast the amount of time needed to complete the experiment and report back the estimated time to completion. The user would have the option to stop the execution at any time and look at the results up to that point and then terminate the run or continue. At the completion of the experiment, the system would perform the needed statistical analysis based upon the desired result and particular set of circumstances (presence or absence of autocorrelation, type of experiment etc.). The system would then report the results in natural language and ask the user if graphical results are desired.

SUMMARY:

As we watch the latest developments in the areas of simulation methodology, simulation language developments, computer graphics, computer hardware, and expert systems research, it is impossible to escape the conclusion that AI based Expert Simulation Systems will very soon be available. For example in the area of manufacturing we can see the first attempts and steps in this direction (10, 11, 12). It will be an exciting and challenging time. Profound changes are going to come as a result of the efforts in AI and anyone serious about the progress of their organizations had best be aware of the potentiallity and inevitability of it.

REFERENCES

1. Turing, A.M., "Computing Machinery and Intelligence," Mind, Vol LIX, Oct. 1950, pp:433-60.

2. McCorduck, Pamela, Machines Who Think, W.H. Freeman and Co., San Francisco, 1979.

3. Williamson, Mickey, "Artificial Intelligence: Expert Systems Can Turn Your PC Into an Intelligent Assistant," PC Week, Vol. 1 No. 24, June 19,1984.

4. Young, R.E. and R. Mayer, "History and Introduction to Artificial Intelligence and Expert Systems," Working Paper, Industrial Automation Laboratory, Industrial Engineering Dept., Texas A&M University, April 1984.

5. Feigenbaum, E., et al, "Generality and Problem Solving: A Case Study Using the DENDRAL Program," Machine Intelligence,Vol. 6, 1971, pp:165-90.

6. Davis, R., et al, "Production Rules as a Representation for Knowledge-Based Consultation Program," Artificial Intelligence, Vol. 8, No. 1, 1977, pp:14-45.

7. Stallman, R.M. and G.J. Sussman, "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis," Artificial Intelligence, Vol. 9,19 77, pp:135-96.

8. Stefic, M., et al, "The Organization of Expert Systems: A Prescriptive Tutorial, "Artificial Intelligence,Vol.18 , No.2, Mar. 1982, pp:135-73.

9. Michie, Donald, "Knowledge-based Systems," University of Illinois at Urbana-Champaign, Report 80-1001, Jan.198 0.

10. Vere, S. A., " Planning in Time: Windows and Durations for Activities and Goals," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 5, No. 3, May 1983, pp:246 -267.

11. Fox, M.A., "The Intelligent Management System: An Overview," Carnegie-Mellon University Report No. CMV-RI-TR-81-4, DECEMBER 1982.

12. Bullers, W.I., et al, "Artificial Intelligence in Manufacturing Planning and Control," AIIE Transactions, Vol. 12, No. 4, December 1980, pp:351-363.