AI-The Three Toed Sloth

Robert A. Karlin
Karlins' Korner
7628 Van Noord Ave.
N. Hollywood Ca.
91605

The three toed sloth, the ai, is a large slow beast that lives in the South American jungles, where he hangs upside down from the trees, feeding on fruits and vegetables. Though slow and normally docile, the ai is a powerful animal, and can be dangerous when aroused.

Artificial Intelligence, that is AI, is also a large slow beast, living far from the ken of normal programmers. Close observation of AI by programmers more used to the mundane could easily lead them to believe that the AI has been programmed upside down, or at least by creatures that live in trees, feeding on fruits and vegetables. And AI, though slow and normally docile, can indeed be very powerful, and also very dangerous if misused.

INTRODUCTION

Since the beginning of the computer age, man has wondered at the idea of a thinking machine. From the Dybbuk of eastern European myth to the clockwork figures of seventeenth and eighteenth century fiction, thinking machines were usually envisioned as human in shape. In 1923, Karel Capek coined the word Robot in his play, *R.U.R.*, and since then, the word has been synonymous with mechanical intelligence. It wasn't until the creation of Eniac and Univac, the first commercial computing machines, in the mid 1950s that the actual mechanism by which intelligence could be imparted to nonliving structures took shape. Since that time, the search for artificial intelligence, has been ceaseless.

In general, AI has not touched the business market place. AI is still too new a field of study to have produced much fruit. However, this is changing. This paper is a look at AI from the businessman's point of view, examining what has gone before, and what is still to come, and how this will affect the business data processing department of the near future.

INTELLIGENCE

Before we enter into a discussion of what constitutes artificial intelligence, we should first see if we can define what it is that we mean by intelligence.

Webster's defines intelligence as 'the ability to learn or understand from experience; the ability to acquire and retain knowledge; the ability to respond quickly and successfully to a new situation'. What part of this definition is applicable to intelligence of the artificial kind?

By far the most striking difference between computer programs and human beings is the ability of humans to alter their behavior pattern based on experience. But it is possible to create programs that learn as well. The simplest example of a program that learns is the game program ANIMAL, a computerized version of twenty questions. To initiate the game, the program says 'THINK OF AN ANIMAL'. After the player responds with a carraige return, the program inquires 'DOES THIS ANIMAL HAVE FOUR FEET?'. If you respond in the affirmative, the program will say 'ARE YOU THINKING OF A CAT?'. If you respond that you are not thinking of a cat, the program will ask you what animal you are thinking of, and when you respond 'elephant', the program will ask for an indicative feature of an elephant. The next time the game is played, if the answer to the first question is affirmative, the program will ask the question saved from the first game to differentiate a cat from an elephant.

If the program again guesses wrong, it will store the information acquired this play. In a surprisingly short time, the program can accurately guess thousands of separate animals, usually with less than ten questions. This form of trial by error can be very effective for small problems, but almost worthless for anything larger. To illustrate, imagine a program that plays chess. After each loss, the software stores the last move prior to the mate, and eliminates it from its possible moves. To develop any coherent play in this manner would take years, even with our fastest machines, and the lookup time during play would be prohibitive. Current research in the area of software learning is concentrating on methods for deriving underlying rules of thumb, as opposed to specific courses of action.

It may seem that acquisition and retention of knowledge is the easiest segment of intelligence to emulate in machinery. After all, this is what we believe computers do best, storing data. And yet, just storing data does not really fit what Webster was driving at. Data must be stored in some usable form, indexed appropriately for later retrieval, and summarized into coherent structures. And this is where we run into trouble. We can store data much more easily than we can produce general purpose rules to identify and classify that data. We have trouble developing rules to allow a program the ability to distinguish between a photo of a beach ball and a photo of the sun. We also have trouble eliminating "noise" data, data that does not belong, from data that just does not fit. A human can easily look into a basket of objects and retrieve a particular size and color block, yet there is no program yet that can perform that

task with one hundred percent accuracy.

The ability to respond to new situations based on prior experience may seem to be the most difficult section of our definition to emulate, yet is actually one of the success stories of current AI research. Rule-based 'expert systems' have been developed that can generalize from insufficient data and, responding to new situations with its area of expertise, produce fairly reliable results, but the key to a successful expert system seems to be as much in the artistry of the design analyst as in the developmental technique. We have yet to produce an expert system able to design other expert systems, though this project is certainly being pursued.

EXPERT SYSTEMS

The biggest success story of artificial intelligence research is knowledge-based or 'expert' systems. The first program that could be called knowledge-based was called DENDRAL, and was developed at Stanford University in the mid 1960s in order to help chemists identify compounds by spectrometric analysis. Since that time, expert systems have been designed for applications as diverse as oil prospecting and medicine. In addition to complete applications, expert system 'shells' are now available, allowing customers to tailor the system to their own needs.

All expert systems contain at least two basic parts. First, obviously is the 'knowledge base' itself and second, some form of knowledge interpreter to input and retrieve data from the knowledge base.

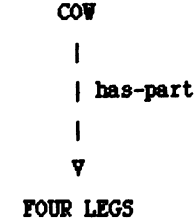
In order to represent a field of knowledge, it must be codified in a way that will make it both accessible and understandable. Some of the different ways of coding are logical coding, procedural representation, semantic nets, production systems, and frames.

Logical coding consists of formal logical statements. For example, if we take the statement 'all cows have four legs', then we could express this in formal logical expression as 'For any object x, if x is a cow, then x has four legs'. The advantage of logical coding is that the rules by which expressions are evaluated is based

on centuries of philosophical research, and is known and well understood.

Procedural representation consists of small well defined procedures that process individual portions of the problem. For example, if we were building a natural language parsing program, nouns, verbs, adverbs, etc. each would have their own small procedure that determines what actions should be taken. The major disadvantage to procedural based systems is their complexity, creating problems in understanding the entire interaction of the base procedures and making debugging difficult.

Semantic nets most resemble the database of the business community. Objects, concepts and events are stored as 'nodes', and the interrelation of these nodes are stored as 'links'. A simple net might be:



The major problem with nets is that they are not innately valid, that is, the interpretation of what a link means is entirely in the hands of the software, and, unlike the logical representation, the data itself is no guarantee of its meaning.

Productions systems, also known as rule-based systems, store information as a set of rules, called productions, usually in the form of 'if condition, then action'. An example may be, 'If it starts raining, and you are outside, then open umbrella'. Because of their innate understandability, production systems have been useful for large application systems. DENDRAL, mentioned above, and PROSPECTOR, a geological program, are among the better known of these.

The last representational scheme we will discuss is the 'frame', or object oriented representation. Unlike rule based systems, where each 'unit' is procedural, and nets in which each unit is declarative, each frame includes both a declarative and procedural portion. This allows the frame itself to determine what action it should take to any action against the frame. Object oriented coding is becoming popular outside of the AI research centers, and object oriented PASCAL compilers are even now becoming available.

In addition to our knowledge base, we must have some form of program to analyze our queries and convert them into some form of database access. Each different form of data storage has its own type of 'inference engine' to form the bridge between the user and the data and to represent the methodology of the original expert. Much of the research in this area centers around the ability to control and predict how the system will function under a broad range of circumstances. These techniques are slowly working their way

into the business community to produce management reporting and long range data analysis and prediction.

Expert systems have been successful, when they are, due to the narrowness of the scope chosen for each project. Knowledge based systems are massive undertakings, involving skilled 'knowledge engineers', who are responsible for interpreting the methods of each expert chosen for a system model, and converting this knowledge and methodology into a program and database. As any analyst knows, even the expert may not know what he is doing when he exercises his talent, so separating the substance from the window dressing can be quite an undertaking. Even after the system is complete, improved technology and current information must still be added continuously to keep the system from becoming obsolete. And yet, expert systems can pay for themselves in a single use, capturing an expertise that would be lost forever.

Even though we may seem to have produced the embryo of machine intelligence, we are beginning to realize that humans think differently than the models we have built. Let us look at how researchers have tried to provide machine intelligence with the ability to solve problems.

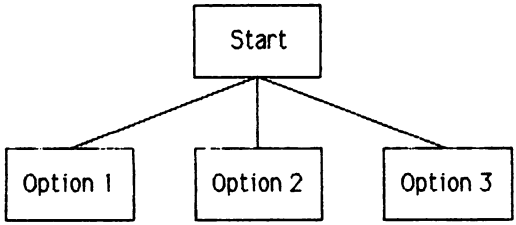
PROBLEM SOLVING

The area of game playing has been one of the most successful areas of research in AI. Most of the leaders of AI were fascinated with games research, due not just to the enjoyment of game playing, but to the limited domain that exist in a game environment, allowing a game simulation to act as a proxy for more complicated real world problems. One of the most popular game simulations has been, of course, chess. The first paper on the subject, 'Programming a Computer for Playing Chess', was published in 1950 by Claude Shannon, and many of the techniques described are used by today's chess playing machines. Chess is a good example of how problem solving techniques have developed.

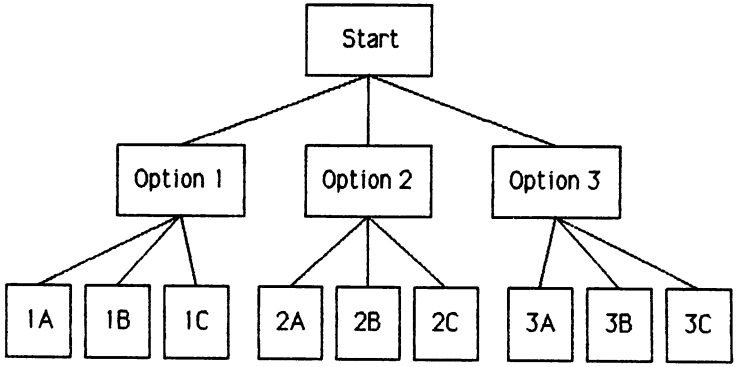
In the beginning, most scholars tended to believe that all that would be needed to develop a good chess program was enough storage and speed to examine all possible moves that could be played for, say, the next ten turns. Most chess experts, by the way, look ahead about six moves. However, if we say that, as an average, we can move at least ten pieces in any one move, and our opponents can move the same, we would need to evaluate 10^{20} moves to look ten moves ahead. Even if we could process a million moves per second, it would take about $3x10^{10}$ hours, or about 3 million years. Obviously, this is not quite acceptable for an afternoon game of chess. Some way must be found to shorten our search. This area of research has been one of the most important areas of AI, and is basic to almost all other AI areas.

In order to examine search techniques, we will represent our problems in the form of a 'search tree'. We start at the top of our tree as so:

From this point, let us say, we have three possible options. We would represent them as so:



Again, let us say that each of these options can have three possible options of their own:



And so forth.

There are a number of ways we can search this tree. The first, and easiest, type of search is called the 'depth-first' search. This, as its name implies, involves searching each branch of the tree to its bottom most leaf, that is, from start to option 1, to 1A, to 1B, to 1C, to option 2, to 2A, etc. Strictly at random, this type of search will, on an average, hit half of the nodes to be searched before finding a 'hit'. The second type of search is known as the 'breadth-first' search. This would involve searching each level, from start to option 1, to option 2, to option 3, to 1A, to 1B, to 1C, to 2A, etc. This search type has the disadvantage of needing a very large intermediate storage area to store the results

of searching each level, and, in a random search, will probably hit a greater number of nodes. On the other hand, breadth-first searches are useful in many game situations, since they allow partial evaluation of each major limb of the tree before continuing the search, and this, in turn would allow the program to pick the limb most likely to contain the solution.

Another technique that helps shorten search time is called 'bidirectional reasoning'. In this type of search, not only do you start a breadth-first search from the top of the tree, you can start a breadth-first search from the goal. This is useful for types of manipulations, such as mazes, in which the goal and its precursors are clearly defined, and we are looking for a relationship between the start and the goal.

Another important problem solving tool is called 'means-ends analysis'. This method involves determining, for each node in the tree, what is required to accomplish the execution of the node. These are then analyzed for each complete path from the top of the tree to the required solution, and an optimum strategy for traversing the tree is evolved.

Most problems in AI involve using a combination of techniques to achieve a solution. The dynamic selection of search algorithms based on the type of problem presented is another continuing area of AI research, and is an area that can be of direct benefit to the business community. As the size and complexity of our data base technologies increase, we will need to include many of the problem

solving methods of AI just to keep the cost of our inquiries manageable.

NATURAL LANGUAGE SYSTEMS

The ability to understand and use language has always been a yardstick for determining intelligence. Even after the discovery that other species used language, this fact was used as evidence of their intelligence, and closeness to man on the evolutionary scale. Even the intelligence of dogs and cats is not usually considered to be their innate ability to solve complex problems in opening doors and proceeding through mazes, but their ability to understand human speech. Is it any wonder that the use of a natural language, such as English, French, or Swahili as opposed to COBOL, PASCAL, or LISP, is considered prime evidence of machine intelligence?

The problems besetting the analysis of natural language can be easily illustrated in the following examples. First, examine the following sentences:

Time flies like an arrow.
Fruit flies like a banana.

In the first sentence, 'flies' is a verb and 'like' is a preposition. In the second, 'flies' is a noun and 'like' is a verb. This type of uncertainty can give software fits. A second example might be: The ball was hit by the boy with the hat. Did the boy hit the ball using a hat, or is the hat just the means by which we should identify the boy? Change 'hat' to 'bat' and answer the question again.

Natural language research was given an enormous boost by the work of one person, American mathematician Naom Chomsky, who, in 1957, revolutionized linguistics with the publishing of 'Syntactical Structures'. Chomsky was the first person to treat grammar as an area of study that transcends specific languages, an area of study with rules common to all languages and able to be expressed in logical and mathematical terminology. Chomsky's rules of grammar allowed researchers to begin to codify the methods by which humans actually decipher language, and to apply some of these techniques to machine intelligence.

As natural language systems were refined, it became obvious that a rule based language system would become complex to the point of impossibility, not just because of the size of the database, but also because of the inability to debug a system of that magnitude. Object based designs and constrained representation systems are now being applied to the problem in order to simplify the basic systems by reducing the number of rules the system must deal with.

Natural language systems are increasing in importance in the business community. As relational databases become prevalent, natural language interface systems are becoming popular, since this allows a user of no computer sophistication to create complex database queries without learning 'computer shorthand' languages, such as SQL. But even the simplest natural language query processor needs an abundance of data storage and CPU cycles. New ideas from AI research are eagerly awaited.

A second area of natural language systems that is becoming popular in commercial applications is language translation programming. It may seem unimportant to be able to translate *Les Miserables* into Russian using a computer, but the ability to translate technical manuals and documents of all sorts will speed the spread of technologies around the world. It may be possible to read technical journals published in Japan or Germany today, instead of waiting weeks for the translations to be completed. Even computer software interfaces could be translated, expanding the markets for these products, and the hardware they are written for.

We have seen that natural language systems are not simple. On the other hand, the economic incentives for working natural language systems seem to be expanding this area of AI at a rapid rate. Combined with the concurrent research in voice synthesis and speech recognition, we may soon be seeing systems that can be queried in English over the phone to tell us the weather in Bankok, or the balance in our checking accounts.

WHERE ARE WE NOW

We have looked at a bit of AI technology, and we have discussed a few of the applications now under study. We have completely ignored the areas of robotics; visual, aural and tactile perception; self programming and self enhancing systems; etc. The purpose of this paper was to give an overview of the field of AI, with emphasis on those areas that may become applicable to the business community of today.

In general, we have seen that AI is usually very costly, both to design and to execute. Many 'commercial' AI systems were originally developed in traditional AI languages, such as LISP, and PROLOG, and later converted to PASCAL, FORTRAN and even COBOL to achieve the speed necessary to compete in the commercial marketplace. As AI moves from a hit and miss field of study to a fully engineered science, more applications will find their way into the market place. Beware, however, the vendor who is trying to sell you an 'expert' system for $495.90. The most charitable point that may be made would be that an overzealous sales force tacked the appellation to the product without looking the words up. On the other hand, this could be another method of separating the unwary from their money. It will be quite a few years before even a reasonable subset could be found at computer boutique prices.

And yet, certain subsets of expert systems and natural language systems are beginning to appear economical. In not to many years,

it will be common to find query languages that seem to be English, because the subset of English implemented will be fairly large. You should expect these programs to be very literal in their interpretation. Asking 'Can you get me the sales figures from October?' may get you an answer of 'YES'. These programs will also tend to get confused often enough for users to comment on the need to desk check the output before treating it as gospel.

We should also see improved database access techniques, with algorithms to optimize inquiries based on prior experience. Our concept of data storage will tend to include concepts such as 'self-defining' databases, whose complete structure, including source, responsibility, editing criteria and report format, will be stored as part of the database itself. We should begin to see object oriented extensions to our current languages. These are extremely useful in rapidly changing businesses, such as life insurance and commodities trading, and would ease the problems of implementing new products or modifying existing ones.

Though we may not see Karel Capek's robots in our offices next week, AI is here to stay, and will be a large part of the office of tomorrow.

BIBLIOGRAPHY

I have found the following books to be extremely interesting while researching this paper. I would highly recommend them to anyone who is interested in the subject of Artificial Intelligence.

THINKING MACHINES The Search for Artificial Intelligence by Igor Aleksander and Piers Burnett. 1987 (Alfred A. Knopf, Inc). This book is an exceedingly clear book describing the area of AI.

The HANDBOOK of ARTIFICIAL INTELLIGENCE volumes 1, 2, and 3 by Avron Barr and Edward A. Feigenbaum. 1981 (William Kaufmann, Inc). The definitive text on artificial intelligence. No study of the field could be complete without it. Again, this book is quite easy to read, and even the most difficult examples are presented with grace and style.

AI in the 1980s and BEYOND an MIT Survey, edited by W. Eric L. Grimson and Ramesh S. Patil. 1987 (Massachusetts Institute of Technology Press). A provocative look at AI present and future.

GODEL, ESCHER, BACH: an Eternal Golden Braid Douglas R. Hofstadter. 1979 (Basic Books, Inc). One of the unique books of our time, GEB is a marvelous blend of treatise and nonsense, exploring the world of artificial intelligence in a manner akin to Lewis Carrol, creating a world of Zen AI. This book is not a quick read, but it is well worth your while to explore it.