Artificial Intelligence

an entry for
The MacMillan Encyclopedia of Computer Science

George F. Luger
&
William A. Stubblefield

Computer Science Department
University of New Mexico

May 1991

A. Definition

Artificial Intelligence (AI) is that branch of computer science concerned with the study of knowledge representation and search. Its goals are the creation of computer programs that behave in an intelligent fashion, and the use of computational models to better understand intelligent behavior in humans, machines, and societies.

In expanding this definition, we examine artificial intelligence from three different perspectives:

1. **Historical.** Artificial intelligence is considered a subdiscipline of computer science. Although the roots of AI include an older tradition of philosophical and psychological efforts to understand the nature of intelligence, AI as a discipline began with the creation of the computer, as seen in Section B.

2. **Methodological.** Traditionally, artificial intelligence has emphasized the use of symbolic data structures to represent knowledge of a domain, formal rules of inference to manipulate these structures and search algorithms to apply this knowledge to problem solving. Section C examines this methodology.

3. **Application.** Progress in AI is driven by a number of difficult and important application areas, including natural language understanding, expert systems, intelligent problem solving, planning and learning. The article concludes, Section D, with an outline of these applications.

B. Computers, the Turing test, and the Origins of AI

Although the 18th, 19th and early 20th centuries saw the formalization of science and mathematics, it wasn't until the creation of the digital computer that AI became a viable scientific discipline. By the end of the 1940's, electronic digital computers demonstrated their potential to provide the memory and processing power required by intelligent programs. It was then first possible to implement formal reasoning systems on a computer and empirically test their sufficiency for exhibiting intelligence.

One of the first papers to address the question of machine intelligence was written in 1950 by the British mathematician Alan Turing. *Computing Machinery and Intelligence* (Turing 1950) is important, both for Turing's conjecture that intelligence is open to a formal characterization, independent of a biological brain, and his response to arguments against the possibility of automating intelligence.

While Turing had already given computing machines a precise characterization through his definition of the *Turing machine,* he recognized that intelligence itself was not amenable to such mathematical axiomatization. In order to circumvent the difficulties of defining intelligence, Turing proposed a behavioral evaluation of intelligent programs, called the *imitation game* and now refered to as *the Turing test.* This test measures the performance of an allegedly intelligent machine against that of a human being. The test places a human and the computer in rooms apart from a second human, the *interrogator.* The interrogator communicates with them by means of a textual device such as a computer terminal and attempts to distinguish the computer from the human on the basis of answers to questions asked over this device. If the interrogator is unable to reliably distinguish the machine from the human, then, Turing argues, the machine must be regarded as intelligent.

Important features of the Turing test are:

1. It gives us an objective notion of intelligence, i.e., the behavior of a known intelligent being (the human) in response to a set of questions. This standard allows us to determine if the machine is intelligent, while avoiding questions about the "true" nature of intelligence.

2. It prevents us from being sidetracked by confusing currently unanswerable questions relating to whether the machine uses the "appropriate internal processes" or whether the computer is actually "conscious."

3. As a blind test, it eliminates any bias in favor of the human over the machine by forcing the interrogator to focus entirely on the content of the questions.

The Turing test has played an important role in the development of AI. Besides giving theoreticians a criteria for recognizing an intelligent program, it also gives applications designers a basis for evaluating software performance. Expert systems, for example, are usually validated by comparing the performance of the program to that of a human expert on a common set of problem instances. The MYCIN program, for instance, was evaluated by letting a group of practitioners blindly evaluate the diagnoses provided by MYCIN and a group of human practitioners on ten actual meningitis cases (Buchanan and Shortliffe 1984).

AI is unique among the sciences in tracing its beginnings as a formal discipline to a specific place and time. The Dartmouth Summer Research Project on Artificial Intelligence was held in 1956, and was the first gathering of researchers who had written computer programs exhibiting some form of intelligent behavior. A brief description of several of the invited presentations at this workshop and other early research provides a perspective on the state of the art in AI in the late 50's and early 60's. This work is also important in that it established many of the core concerns of contemporary artificial intelligence.

Arthur Samuel's checker player (Samuel 1959) used a number of techniques first developed in the field of operations research. These include *hill climbing, mini-max search,* and the use of an evaluation polynomial to rate board states. This program was remarkable in a number of ways: it was the first example of a machine learning program, improving its performance by changing its evaluation function and by storing board states and their evaluations for future use. Through these techniques, the program learned to play an excellent game of checkers; more importantly, the techniques Samuel developed remain a vital part of machine learning methodology. In addition, the program was impressive for its use of sophisticated representational and memory management techniques to overcome the limitations of 1950's hardware.

Herbert Gelernter (Gelernter 1963) developed a geometry theorem proving program that successfully proved many theorems from high school geometry texts.

Newell, Shaw and Simon, then researchers at the Carnegie Institute of Technology, presented their *Logic Theorist* (Newell and Simon 1963a). They also presented some interesting research comparing a run of their computer program proving mathematical theorems with traces of human subjects solving the same theorems. This comparative research is arguably the beginning of work in cognitive science, the use of computer programs as a tool for modelling human cognition. EPAM (Feigenbaum 1963) was another early cognitive program which attempted to model the way in which humans memorize and index sets of nonsense syllables. EPAM was developed by Edward Feigenbaum, one of Herbert Simon's students; Feigenbaum was later to play an important role in the development of the first expert systems and the Stanford AI research laboratory.

Newell and Simon's *General Problem Solver* (Newell and Simon 1963b) was the first effort to provide a unified theory of human cognition in the form of a computer program. The GPS assumed that any problem could be solved by a general method of heuristic state space search.

It was also at the Dartmouth conference that John McCarthy selected the name *Artificial Intelligence* for the new field of research. The word *artificial* was selected for its literal meaning: to make (the Latin verb *facere*) by skilled effort (*ars, artis*, the root for the English *artist* or *artisan*). McCarthy's early contribution to AI was the design of the LISP (LISt Processing) language. Over the years, LISP has been the single most important tool for AI research and development.

These early results were collected in the book, *Computers and Thought* (Feigenbaum and Feldman 1963). The continuing royalties from this collection still finds the *Computers and Thought Award*, presented biannually at the International Joint Conference on Artificial Intelligence (IJCAI). Technical conferences such as IJCAI, and the yearly American Association for Artificial Intelligence (AAAI) conference continue to serve as the primary forums for research in artificial intelligence. In addition, communication among AI researchers has been promoted by the numerous books and technical journals including *Artificial Intelligence, The AI Magazine, Computational Intelligence* and *Cognitive Science.*

C. The methodology of Modern Artificial Intelligence

If there is any unifying principle to current work in artificial intelligence, it is the *physical symbol system hypothesis* (Newell and Simon 1976):

> A physical symbol system consists of a set of entities, called symbols, which are physical patterns that can occur as components of another type of entity called and expression (or symbol structure). Thus a symbol structure is composed of a number of instances (or tokens) of symbols related in some physical way (such as one token being next to another). At any instant of time, the system will contain a collection of these symbol structures. Besides these

structures, the system also contains a collection of processes that operate on expressions to produce other expressions: processes of creation, modification, reproduction and destruction. A physical symbol system is a machine that produces through time an evolving collection of symbol structures. Such a system exists in a world of objects wider than just those symbol expressions themselves. (p. 116)

Newell and Simon then state the physical symbol system hypothesis:

A physical symbol system has the necessary and sufficient means for general intelligent action.

This hypothesis states the underlying assumptions of AI in a succinct and empirically falsifiable form. In addition, it outlines the methodology of "mainstream" work in artificial intelligence:

1. Represent the important objects and relationships in the problem domain as sentences in an appropriate formal language.

2. Define operations which transform these sentences in ways that reflect the semantics of the problem domain. For example, these operations may be atomic actions that a robot may take in the world, or rules in an expert system or logical reasoning system.

3. Problem instances consist of a description (in the language of step 1) of the given data or starting conditions of the instance (the start state) and a description of the desired solution (the goal state).

4. Search the space of sentences produced by applying these operations to legal expressions to find a series of operations that lead from the starting state to the goal state.

The next section will examine the implementation of this methodology in more detail.

C.1 Problem Solving and State Space Search

State space searcg is based on the assumption that any problem could be solved by searching through a space of appropriately defined operations on formal sentences. These operations define a space of partial solutions called the *state space*. The technique of solving problems by searching this space is called state space search: the search for a series of operations that transform the starting state into the required goal state.

Much early work (1960 through the mid 1970's) in AI was devoted to developing state space search as a general problem solving method. This methodology was successfully applied to a variety of fields including mathematical theorem proving, planning, and problem solving. This work focused on finding languages that could adequately describe classes of problems and strategies for efficiently searching the resulting space. Many of these strategies, such as means ends analysis, were *heuristic* in nature. That is, they provided intelligent guidance through the space while not guaranteeing that such guidance would always produce an optimal problem solution. Heuristics are further examined in section C.2.

In developing the methodology of state space search, the first order predicate calculus was often chosen as a general language for describing problem states. First order predicate calculus, because of its roots in mathematics and formal logic, provides well understood, logically sound and complete inference rules. It can also be shown that predicate calculus based theorem provers are *turing complete;* that is, any process that could in principle be formalized can be formalized using predicate calculus. This further strengthened its appeal as a general language for intelligent systems.

For example, suppose we would like to apply the methodology of state space search to find a series of actions to transform a starting state into a desired goal (Figure 1).

****insert Figure 1 here*****

The start state of the world is described using predicates:

block(a).	block(b).	block(c).
clear(a).	clear(c).	ontable(b).
ontable(c).	on(a,b).	inhand(nil).

Similarly, we describe the goal state:

block(a).	block(b).	block(c).
clear(a).	clear(c).	ontable(b).
ontable(a).	on(c,b).	inhand(nil).

The operations that the robot can perform are also described using predicate calculus. These operations consist of *preconditions*, which define circumstances when the operation may be performed; an *add list* of sentences that are true when the operation completes; and a *delete list* of things that are no longer true when the operation completes. Some of the operations given our block stacking robot include:

pickup(X):
  Preconditions: clear(X) $\wedge$ inhand(nil)
  Delete list: clear(X) $\wedge$ on(X, Y)
  Add list: inhand(X)

stack(X,Y)
  Preconditions: clear(Y) $\wedge$ inhand(X)
  Delete list: clear(Y) $\wedge$ inhand(X)
  Add list: inhand(nil) $\wedge$ on(X, Y)

In addition, the planner has rules for inferring additional properties of blocks world descriptions. Such a rule might allow the program to infer when a block is clear:

$$\forall X \text{ block}(X) \wedge (\neg\exists Y \text{ block}(Y) \wedge \text{on}(Y, X)) \rightarrow \text{clear}(X)$$

This rule states that for all X, if X is a block and there does not exist a block, Y, such that Y is on X, then X is clear. Note that much of the power of predicate calculus comes from the use of variables (X, Y) to define general rules and operations such as that given above.

Given this description of the blocks world domain, we may solve problems by searching the state space to find a series of moves that transform the beginning state into the goal state, as in Figure 2.

****Insert Figure 2 here****

As this example shows, search based problem solving may be adapted to a variety of problem domains. By establishing the methodology of knowledge representation and state space search, this early work provided a framework for artificial intelligence. This research explored a number of search strategies that continue to be used today, including *forward search,* in which problem solving begins with data and searches forward to find a solution, and *backward search,* in which the search process begins with the desired goal and works back to the given data. Other strategies explored at this time include *depth first search, breadth first search* and *best first search.* These strategies are described in more detail in (Luger and Stubblefield 1989).

C.2 Production Systems

Another important AI technique to emerge from this early research was the *production system architecture* , Figure 3. The production system is a control method that has proven particularly important for implementing search algorithms, designing expert system building tools and for modelling human performance. The production system provides pattern directed control for an inferencing system; a rule is used when and if it matches a state of the problem solving process. A production system consists of three components: a set of *production rules*, a *working memory* and a *recognize-act control cycle.*

1. The set of production rules. A production is a condition- action pair that describes a single chunk of problem solving knowledge. The condition part of the rule defines when the rule matches a state of a problem solving process. When the condition matches a problem state, the rule may be *fired*, performing the associated action. Production rules are often represented in an "if...then..." syntax.

2. Working memory contains a description of the current state of the problem solving process. The contents of working memory are matched against the conditions of the production rules, producing a set of *enabled* productions.

3. The recognize-act cycle is straightforward: the current state of problem solving is maintained as a set of patterns in working memory. These patterns are matched against the rules to produce a set of enabled rules, called the *conflict set. Conflict resolution* is the process by which the production system selects a rule from this set. This rule is then *fired.* Its action modifies the contents of working memory, and the cycle repeats. This continues until recognition produces an empty conflict set. At this point the system stops and the problem solution is in working memory.

****insert Figure 3 here****

Newell and Simon used production systems to model human problem solving in pioneering research done at Carnegie Mellon University in the 1960's and 70's. In their approach, production rules represented the knowledge and skills a human being might possess. These might be rules for playing chess, solving physics problems or diagnosing diseases. Working memory represents the human's short term memory or attention at a given time in the problem solving process. Thus, the production system may be interpreted as modelling the activation of skills or chunks of knowledge by the human's current focus of attention, and the application of those skills to problem solving.

The production system has proven an effective architecture for the implementation of AI programs for a number of reasons, including:

1. **Ease of implementing state space search.** The successive configurations of working memory correspond to neighboring states in a space. The firing of rules implements a transition between states.

2. **Pattern driven reasoning.** This simplifies the implementation of opportunistic problem solving strategies, as required by AI. It allows a highly flexible control structure in which the needs of a problem situation determine what is done next, rather than relying on a previously specified procedural ordering.

3. **Representational flexibility.** We may use any language to represent production rules, just as long as it supports pattern matching. For example, a production system could be used to implement the planner of the previous section.

4. **Flexibility of control.** The production system supports a variety of search strategies. By placing the given data of a problem in working memory and matching conditions against this data, the reasoning proceeds in a forward fashion. By placing a goal in working memory and matching against rule conclusions, the production system implements backwards reasoning. In addition, by modifying the conflict resolution strategy, the system can implement a variety of heuristic based strategies.

For more information on production systems see (Luger and Stubblefield 1989; Waterman and Hayes-Roth 1978)

C.3 Strong Methods and the Rise of Knowledge Based Systems

The major focus of this middle period (1960 through the early 1970s) of AI research was on the development of general architectures for intelligent problem solving. The production system is an example of this orientation. Another focus of this work was on the use of *heuristics* to control search. Search based problem solving is plagued by the large number of states in interesting problem spaces. As a search algorithm moves into a space, the number of states tends to increase exponentially. Consequently, intelligent problem solvers must have some way of effectively guiding the search process.

A heuristic is any problem solving method which improves the efficiency of a problem solver(Pearl 1984); in a production system, for example, a heuristic may be used to select a rule in conflict resolution. However, this increase in efficiency has a price, heuristic search may sacrifice the quality of solutions or even the ability of the problem solver to find a solution to every instance. Means ends analysis, as discussed in C.1 is an example of a heuristic. While it is effective in choosing a operation that reduces the syntactic difference between the current state and the goal, it may break down in certain situations. Consider a situation in which the current problem state is such that any solution must temporarily *increase* the difference between this state and the goal. Means ends analysis cannot select an operator in these situations.

During this middle period, the emphasis was toward finding general principles of intelligent behavior. Consequently, there was much work on finding heuristics, like means ends analysis, that might prove effective across a variety of domains. These heuristics do not use knowledge of a specific domain to guide search, instead, they examine syntactic properties of the representations. Means ends analysis does this by reducing the syntactic differences between states. Another general heuristic might favor production rules that reduce the number of patterns in working memory.

However, research showed that general, syntactic methods, called *weak methods,* were limited in their ability to solve problems. In trying to extend search based problem solving methods to domains of human expertise, such as organic chemistry or medical diagnosis, researchers realized that humans relied upon knowledge that was specific to those domains. These heuristics are called *strong methods,* and underlie the development of *knowledge based systems,* also known as *expert systems.*

MYCIN (Buchanan and Shortliffe 1984), a knowledge based system for diagnosing bacteremia and spinal meningitis, is an early example of a knowledge based system. Rather than relying on weak methods, the designers of MYCIN emphasized the construction of a large base of specific diagnostic knowledge. This knowledge base was represented as "if ... then..." rules. A typical MYCIN rule is

```
if   (a) the infection is primary-bacteremia, and
     (b) the site of the culture is one of the sterile sites, and
     (c) the suspected portal of entry is the gastrointestinal tract
then there is suggestive evidence (0.7) that infection is bacteroid
```

Note that this rule, unlike weak heuristics, is *only* useful for diagnosis of bacteremia. These rules are applied to problem instances by an *inference engine,* essentially a production system, that matches rules with the facts of a specific case.

Knowledge based systems have proven effective in practical domains for a number of reasons. By using AI techniques like pattern directed search, they were able to solve problems that were too complex or irregular to lend themselves to traditional programming techniques. Knowledge based systems are able to explain their reasoning by listing the sequence of rules used in solving a problem; this increases their usefulness and understandability. By applying large amounts of knowledge to problems, they achieved high levels of expertise in a variety of problems that were long considered propriatary to human experts.

Knowledge based systems, based on such general problem solving techniques as state space search, production systems and logical representation, represented an important shift of emphasis for artificial intelligence. Instead of looking for general principles of intelligent behavior, researchers began to focus on methods of acquiring and representing the knowledge used by humans in problem solving. This shift of emphasis has raised a variety of issues that continue to be the focus of ongoing research. How may we determine the specific knowledge used by a human in solving a problem? Humans are notoriously inarticulate about their actual thought processes; consequently, knowledge acquisition remains a difficult problem. Knowledge Based systems explain their reasoning; how may these explanations be best organized and presented? Expert systems, in spite of many current claims, are not human experts; what are the limits of the formalization of knowledge?

Finally, because knowledge based systems require extremely large amounts of knowledge for most domains, researchers have focused a great deal of energy on the development of languages that simplify the acquisition, organization and application of this knowledge. The study of *knowledge representation* has emerged as a central concern of modern artificial intelligence.

Research in knowledge representation is driven by the needs of knowledge intensive problem solvers such as expert systems. Natural language understanding, or the problem of getting machines to understand human languages, has also placed demands on knowledge representation. Representation languages including logic, rules, frames and objects provide AI programmers with a powerful set of tools for building knowledge bases. These techniques are also formalized in a number of commercial software packages that simplify program development. A particularly powerful class of these programs are *hybrid tools*, which provide the programmer with multiple representation languages, typically rules and frame/objects, different inferencing techniques, such as inheritance or rule based reasoning, multiple search strategies, such as forward and backward search, and a rich set of tools for constructing user interfaces and debugging programs. *Readings in Knowledge Representation* (Brachman and Levesque 1985) provides a valuable introduction.

C.4 Alternative Architectures for Intelligent Programs

While the main thrust of AI research has been toward developing more expressive representations and inference strategies, another school of research has focused on alternatives to the explicit representation of problem solving knowledge. *Parallel Distributed Processing, orPDP,* (Rumelhart and McClelland 1986) refers to models of computation which use large numbers of relatively simple computational units, working in combination, to produce powerful results.

Neural networks are typical of this approach. Patterned after the low level architecture of biological brains, neural networks consist of large networks of artificial neurons. Like biological neurons, these elements are extremely simple: they accept signals from other neurons; these inputs are augmented by weights on the input lines. Each element computes a function of these weighted inputs and, depending on the result of this calculation, either send a signal to other neurons or remain quiet. Through the patterns of connections in the network and the ability of neurons to stimulate or inhibit each other, these systems demonstrate powerful collective behaviors.

Figure 4 illustrates a simple type of neural network called a *perceptron* (Rosenblatt 1962), as it might be used in a computer vision system. The inputs to the network are sets of pixels from the image. The outputs are classifications of the image. Through the system of weights and the pattern of connections, the network is able to compute a mapping from the features of the image onto an appropriate classification.

Figure 4 here**

These systems show promise for complementing representation based AI. For example, since the computation is spread out over many elements, parallel distributed approaches can draw conclusions in spite of the presence of significant amounts of noise in the data. Also, progress has been made in developing algorithms which allow PDP systems to learn from experience by, for example, adjusting the weights on the connections in a neural net.

D. Survey of Application Areas

Artificial Intelligence is driven by a number of important and demanding application areas. While it is impossible to be exhaustive in the scope of a single encyclopedia entry, a brief survey of these applications is essential to any discussion of AI.

Automatic Theorem Proving (Wos and others 1984) is one of the oldest areas of AI, tracing its origins past Newell and Simon's *Logic Theorist* to the philosophical work of Russell and Whitehead. Work in theorem proving continues, focusing on the development of more sophisticated models of mathematical reasoning and on extending techniques to a wider range of application areas. In particular, theorem proving has found success in the design and validation of logic circuits and the verification of computer programs.

Knowledge Based Systems (KBS) (Luger and Stubblefield 1989; Waterman 1986) include some of the most successful applications of AI techniques. KBSs have been written in fields as diverse as medical diagnoses, industrial process control, geology and engineering design. Current issues in knowledge based systems include knowledge representation, knowledge acquisition, and the design of intelligent user interfaces.

Planning (Fikes and others 1972) addresses the problem of finding a series of actions that will reach some goal. While robotics are the most obvious application of planning technology, these techniques have also found application in such areas as the design of manufacturing processes and the planning of scientific experiments. Planning is complicated by the necessity of representing changes in a world over time. Also, since such applications as robotics require interaction with the real world, planners must deal with noise and error, allowing for the detection and correction of plan failures. Another important issue for planners is the *frame problem*: how can a planner anticipate the side effects of a given action?

Natural Language Understanding (Allen 1987; Winograd 1983) is one of the most challenging areas of AI research. The goal of creating computer programs that can effectively communicate in human language is an old one. While there have been impressive successes in highly constrained areas of discourse, general language understanding remains elusive. The problem is with the large amounts of knowledge required to understand even simple sentences. Consequently, natural language understanding has driven, and continues to motivate research in knowledge representation and the semantics of human language.

Machine Learning (Kodratoff and Michalski 1990; Michalski and others 1983; Michalski and others 1986) addresses the problem of creating computer programs that can learn, either from their own experience, observation or interpreting high level advice. In addition to work in Parallel Distributed Processing, considerable work has been done in symbol based learning, focusing on the use of prior knowledge to guide learning in new domains.

Knowledge Representation and Reasoning (Brachman and Levesque 1985) continues to be a vital area of research, focusing on the development of representation languages that capture the full range of knowledge, and the development of sophisticated models of representational semantics.

Cognitive Science (Collins and Smith 1988; Newell and Simon 1972) continues with the goal of using computer programs as a tool and a metaphor for understanding human cognition. This field blends AI with psychology, philosophy, linguistics, neuroscience, anthropology and other fields in an effort to understand the functioning of the human mind at neural, cognitive and social levels.

Other important application areas include commonsense reasoning and naive physics, game playing, computer vision, intelligent tutoring systems, the design of intelligent user interfaces, automatic programming, the design of symbolic programming languages and the simulation of complex systems. All of these areas are the focus of active, ongoing research and promise to enrich the field of artificial intelligence for years to come.

E. The Future of AI: Knowledge Media and Autonomous Agents

The effort to produce intelligent problem solvers has led to the development of knowledge based systems: programs which use explicitly represented knowledge to solve problems and explain the reasoning that led to those solutions. This approach has proven its viability through a number of successful programs; however, a number of limitations and unfulfilled promises remain. The difficulties in acquiring and organizing sufficient knowledge, along with the likely impossibility of ever proving that such a knowledge base is actually correct, suggest that we should emphasize the creation of intelligent apprentices and assistants, rather than autonomous intelligent agents. Some researchers, such as Weizenbaum, have questioned the morality of letting machines make decisions in such intimately human domains as law, politics and business (Weizenbaum 1976).

Thus, the current generation of intelligent machines may be seen as extending, rather than replacing our own intellects. However, work will continue on designing machines that are capable of functioning as independent autonomous agents. A range of applications, such as space exploration or the use of robots to perform tasks, like nuclear reactor maintenance, which are too dangerous for humans, could benefit greatly from the development of such machines.

Another path of development accepts the possibility that computers may be inherently limited in their ability to achieve human like intelligence and focuses on the development of intelligent assistants. One of the exciting and revolutionary potential outcomes of this research is the development and growth of electronic knowledge media.

While most contemporary electronic media is highly efficient at storing large amounts of factual information, for example in data bases, they are limited in capturing knowledge. Knowledge requires complex organization: representations such as frames, objects and rules can capture the complex structure of human knowledge. Knowledge is dynamic: an expert, either a human or a machine, can apply the appropriate expertise to specific problem instances. Knowledge is flexible: it can be applied freely to a class of problems. Knowledge changes: knowledge based systems allow frequent updates through the addition of new rules to the knowledge base.

Knowledge base technologies promise the development of media for storing, transmitting, searching, and applying dynamic knowledge, just as books allow us to transmit more static information. As knowledge representation techniques progress and standards evolve, we will see formal knowledge become a marketable commodity. It is possible to envision the development of *knowledge utilities*, supplying knowledge bases to subscribers, much like contemporary utilities supply water and power. These changes in the transmission and use of knowledge will provide science, business and government with powerful tools for progress, as well as dangerous opportunities for their abuse.

Artificial intelligence reflects some of the oldest concerns of western civilization in the light of modern computational technology. The notions of rationality, representation and reason are now under scrutiny as perhaps never before, since computer scientists demand to understand them operationally, even algorithmically. At the same time, the political, economic and ethical situation of our species forces us to confront our responsibility for the effects of our artifices. The interplay between applications and the more humanistic inspirations for much of AI continues to inspire hosts of rich, challenging questions.

A number of general references for AI include *The Encyclopedia of Artificial Intelligence* (Shapiro 1987) and *The Handbook of Artificial Intelligence* (Barr and Feigenbaum 1981). Our discussion of the historical foundation of AI is taken from *Artificial Intelligence and The Design of Expert Systems* (Luger and Stubblefield 1989), where a more detailed discussion of the algorithms and representations of artificial intelligence is available.

References

Allen, J. 1987. *Natural Language Understanding.* Menlo Park, Cal: Benjamin Cummings.

Barr, Avron and E. A. Feigenbaum. 1981. *The Handbook of Artificial Intelligence.* Los Altos, Cal.: William Kaufmann.

Brachman, R. J. and H. J. Levesque, ed. 1985. *Readings in Knowledge Representation.* Los Altos, Cal.: Morgan Kaufmann.

Buchanan, Bruce G. and Edwared H. Shortliffe, ed. 1984. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Reading, Mass.: Addison-Wesley Publishing Company.

Collins, A. and E. E. Smith, ed. 1988. *Readings in Cognitive Science.* Los Altos, Cal.: Morgan Kaufmann.

Feigenbaum, E. A. 1963. The Simulation of Verbal Learning Behavior. In *Computers and Thought,* ed. E. A. Feigenbaum and J. Feldman. New York: McGraw-Hill.

Feigenbaum, E. A. and J. Feldman, ed. 1963. *Computers and Thought.* New York: McGraw-Hill.

Fikes, R., P. Hart, and N. Nilsson. 1972. Learning and Executing Generalized Robot Plans. *Artificial Intelligence* 3 (4): 251-288.

Gelernter, H. 1963. Realization of a Geometry Theorem Proving Machine. In *Computers and Thought,* ed. E. A. Feigenbaum and J. Feldman. New York: McGraw-Hill.

Kodratoff, Y. and R. S. Michalski, ed. 1990. *Machine Learning: An Artificial Intelligence Approach, Vol III.* Los Altos, Cal.: Morgan Kaufmann.

Luger, George F. and William A. Stubblefield. 1989. *Artificial Intelligence and the Design of Expert Systems.* Redwood City: Benjamin Cummings.

Michalski, R. S., J. G. Carbonell, and T. M. Mitchell, ed. 1983. *Machine Learning: An Artificial Intelligence Approach.* Palo Alto, Cal.: Tioga.

Michalski, R. S., J. G. Carbonell, and T. M. Mitchell, ed. 1986. *Machine Learning: An Artificial Intelligence Approach, Vol II.* Los Altos, Cal.: Morgan Kaufmann.

Newell, A. and H. Simon. 1972. *Human Problem Solving.* Englewood Cliffs, New Jersey: Prentice-Hall.

Newell, A. and H. Simon. 1976. Computer science as empirical inquiry: Symbols and search. *CACM* 19 (3): 113-126.

Newell, A. and H. A. Simon. 1963a. Empirical explorations with the Logic Theory Machine: A case study in heuristics. In *Computers and Thought*, ed. E. A. Feigenbaum and J. Feldman. New York: McGraw-Hill.

Newell, A. and H. A. Simon. 1963b. GPS: A program that simulates human thought. In *Computers and Thought*, ed. E. A. Feigenbaum and J. Feldman. New York: McGraw-Hill.

Pearl, Judea. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Reading, Mass.: Addison-Wesley.

Rosenblatt, F. 1962. *Principles of Neurodynamics.* New York: Spartan.

Rumelhart, D. E. and J. L. McClelland. 1986. *Parallel Distributed Processing.* Cambridge Mass.: MIT Press.

Samuel, A. L. 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development* 11 : 601-617.

Shapiro, S. C., ed. 1987. *Encyclopedia of Artificial Intelligence.* New York: John Wiley & Sons.

Turing, A. A. 1950. Computing Machinery and Intelligence. *Mind* 59 : 433-460.

Waterman, D. A. 1986. *A Guide to Expert Systems.* Reading, Mass.: Addison Wesley.

Waterman, D. A. and F. Hayes-Roth, ed. 1978. *Pattern Directed Inference Systems.* New York: Academic Press.

Weizenbaum, J. 1976. *Computer Power and Human Reason.* San Francisco: W. H. Freeman.

Winograd, T. 1983. *Language as a Cognitive Process: Syntax.* Reading, Mass: Addison Wesley.

Wos, L., R. Overbeek, E. Lusk, and R. Boyle. 1984. *Automated Reasoning: Introduction and Applications.* Englewood Cliffs, New Jersey: Prentice-Hall.

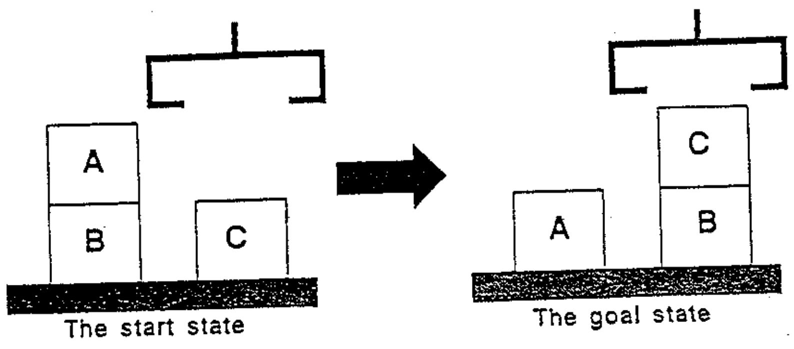

Figure 1. The *start* state and the *goal* state of the "blocks world" problem.

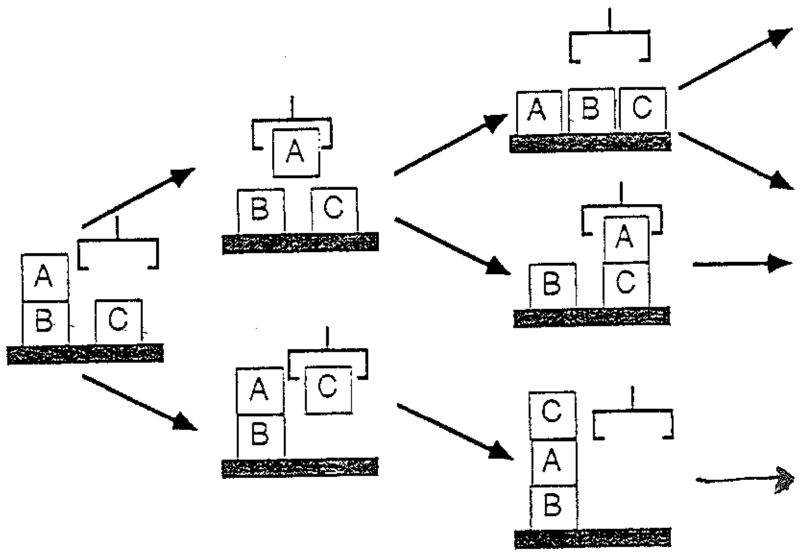

Figure 2. Part of the search space of the "blocks world" problem.

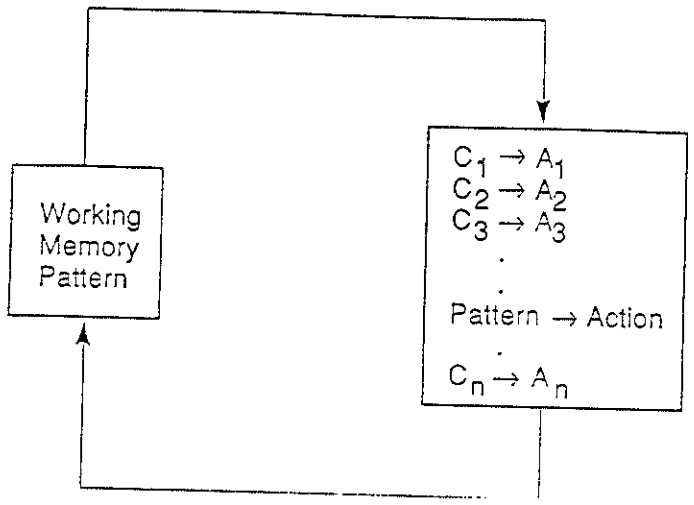

Figure .3 A production system. Control loops until working memory pattern no longer matches the conditions of any productions.

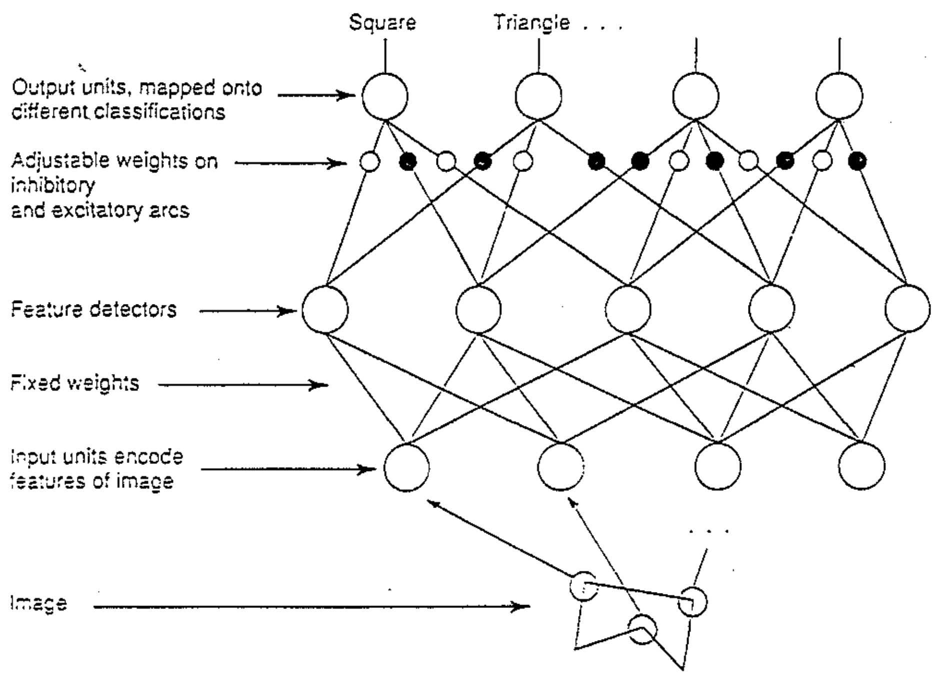
Square
Triangle . . .
Output units, mapped onto different classifications
Adjustable weights on inhibitory and excitatory arcs
Feature detectors
Fixed weights
Input units encode features of image
Image
. . .

Figure 4. An example of a *perceptron* for image classification.