The Relationship between Artificial Intelligence and Command and Control

by

Dennis Kibler

Technical Report #153

Department of Information and Computer Science
University of California, Irvine
Irvine, California 92717

June 1980

Introduction

This paper has a number of goals. It will give a high level, descriptive tutorial of those areas of AI which might have an impact on command and control operations. This outline will describe the state-of-the-art technology, the difficulties, the problems being attacked, and their relationship to C2. It will also make recommendations on how the Navy might stimulate effort in areas most needed by a C2 system. The areas in AI are covered with respect to their importance to C2, as opposed to their importance to AI.

There are a number of relevant concerns which this paper will not regard. For example, assume, for the moment, that all the necessary AI techniques for C2 are available. There would still remain the problems of incorporating such a system into the social and organizational structures that exist. Maintenance and enhancement would be a greater problem than with more conventional software since few people are knowledgeable in AI technology. Nothing will be said of potential failures due to power surges, cold temperature, faulty machines, sabotage, or a host of other calamities, all of whose eventualities should be contemplated and prepared for.

These gloomy possibilities are only mentionned to indicate that the transference of a "working" AI system into a operational environment will take a great deal of effort

and forethought. This paper only deals with the immediate goal of building an AI system to assist in Command and Control.

The views presented in this paper are my own. A less biased and possibly more accurate appraisal of the relationship between AI and C2 would require taking a survey of AI researchers. Such an approach would require a great deal more man-power, time, and money.

The paper is organized into the following subsections:

1. Rule based system, a general discussion.

2. SIAMMER, a particular rule based system illustrating some of the disadvantages and advantages in using AI technology for C2.

3. Problem Solving and Planning, a generalization of rule based system necessary when rules do not suffice.

4. Natural Language Processing, discussed with a view towards establishing a comfortable user interface.

5. Games, results on game playing are outlined to show how planning is done in a microcosm and to give some perspective on the complexity of the planning process and the degree of sophistication and naivete of current AI technology.

6. Learning, highlighting those aspects that could assist in automatically updating the knowledge representations applicable to C2.

7. Knowledge Representation, outlining some of the myriad of problems and their current status.

8. Conclusions, giving general recommendations on what areas of AI the Navy should support.

Each subsection ends with a statement of the specific relationship of that area of AI to C2.

Rule Based Systems

The first AI system to demonstrate that a computer program could reason at a level comparable to that of trained people was the MYCIN[1] program. This system which did medical diagnosis of blood diseases was based on a rule based architecture, also called a production system[2,3]. There are a variety of views of what constitutes a production system. Each author prefers his own definition. This lack of uniformity and agreement is common in any newly developing science. For the neophyte, this disagreement and confusion is disquieting. For the AI community, research is not strait-jacketed by inappropriate definitions and formalisms.

A rule based system consists of three parts: a collection of rules, a data base to which the rules are applied, and a recipe or algorithm for applying the rules to

the data base. In the most general view, each rule consists of a lefthandside (lhs), which is a list of conditions, and a righthandside (rhs), which is a list of actions. In the most restrictive view, each lhs consist of a string of symbols, while the rhs consists of a replacement string of symbols. Some examples of rules together with the fragment of knowledge to which the rule corresponds, are given below.

Rule	Knowledge
X+0=>X	0 is the additive identity
*ch=>*ches	words ending in ch form their plurals by adding es
(storm X) (in Y X)=>not (merchant X)	A ship in a storm is not likely to be a merchant.

Each of the rules above capture some piece of knowledge. The level of detail of a rule is called the granularity. Defining the appropriate level of granularity is a difficult task.

The second constituent of a rule based system is the data base. The three rules listed above would apply to different data bases and would be helpful in solving different implicit goals. In particular, they would apply to:

Data Base	Goal

algebraic expression	simplification
text	pluralization
relations	message monitoring

In fact, data bases might be parse trees, machine code, english sentences, molecular structures, chemical reactions, list of relations, or any other somewhat formally defined structure. The rules correspond to possible legal manipulations of that structure.

The third constituent of a rule based system is the recipe used to apply the rules. The code corresponding to the particular recipe is called the rule interpreter. In the simplest system rules are kept in a list and continually applied to the data base until no rule can fire (be applied). By choosing rules at random or according to some probability functions, stochasic processes may be modelled. In more sophisticated systems, rules are applied according to some dynamic priority scheme or according to a cost analysis estimate. Despite this variety of choices, rule interpreters have two main varieties: forward chaining and backward chaining.

Rules may be used into two distinct ways. The rule for forming plurals of words ending in ch (*ch=>*ches) can be used to derive the plural of "watch" or to find the singular of the work "matches". When rules are applied in the

direction of the arrow, the rule interpreter is of the forward chaining variety. If the rhs of the rule is matched against a stated goal, then the production system is backward chaining. The MYCIN system is a backward chaining system. It works by assuming the patient could have each disease of some 120 diseases and then backward chains to find supporting evidence. Backward chaining systems are usually more efficient, but they depend on two additional constraints. In a backward chaining system one must know the goal state. Consequently, one could not apply backward chaining to algebraic simplification. Moreover, there must be a way of matching the righthand sides of rules with goals. Since righthand sides, the action part of a rule, are often stated as procedures, this can be very difficult. Forward chaining systems do not suffer from these limitations. Forwarding chaining is useful when all conclusions are desired or when no particular goal is sought. Another minor type of production system is the mixed or hybrid variety, in which both forward and backward chaining is done. Some resolution theorem provers have this characteristic. The control over reasoning from the data (forward chaining) or reasoning from the goals (backward chaining) is not well understood.

Production systems have been partially successful in code optimization[4], code synthesis[5], and stock analysis. They have had uncontestable success in several medical

domains, including the domain of internal medicine[6]. In the area of chemical identification from mass spectrometry data, DENDRAL[7] rivals post-doctoral chemists. Production systems have not had success in the areas of theorem proving, planning, game playing or scene analysis. Theoretically any problem solvable by computer can be done within the production system formalism, but there are characteristics of a problem which make it more suitable for a rule based systems. For a successful production system we find

1. a literature for the domain that organizes the knowledge in a rule-like form
2. domain experts who can express how they process the data
3. a clear definition of the problem or its solution.

The number of rules in production systems that perform at human levels varies from several hundred to several thousand. In some domains, people are unable to express rules. For example, if a clenched fist is quickly opened and you then see a paper clip, by what process of thought did this recognition happen? Rules are usually incomplete when first stated. Domain experts forget to make explicit some "obvious" condition. As the system is used, and its reasoning illustrated, errors are found. Since rules are easy to modify, remove or add, this maintenance is not

costly. A typical error of omission occurs when students write down the rules to capture the legal manipulations of a child's blocks world. Student often will include a rule which states that a block with a clear top can be moved to another block with a clear top, and then the system may try to move a block on top of itself. Properties of a domain which makes it unsuitable for production systems include:

1. no person can do the task.
2. the domain knowledge is voluminous, as is generation of art.
3. the knowledge is subconscious.
4. knowledge is not local or rule-like, but is global.
5. routine repeated inferences are not present.

Production system have two special properties that distinquish them form data base management systems. Each conclusion that the system reaches, either by backward or forward chaining, can be justified by the system. The user can make the entire reasoning process explicit, and in this way, evaluate its results. Through this interaction the user can also correct the system's rules. Moreover the system can assign a confidence to each conclusion and explain the reasons for the particular confidence given.

Although production systems have been used for nearly two decades, they still face a number of unsolved problems. Most of these are fairly technical in nature, so they will not be explained in detail. A primary problem is the control over the number of rule applications. Without a reduction in this number, a combinatorial explosion may occur. Another severe problem is the evaluation of anomalous or contradictory evidence. Doyle[8] has recently made a strong attack on this problem by introducing the idea of dependency directed backtracking within a belief maintenance system.

Command and Control The ramifications of the use of production system for C2 are several. Without an explicit knowledge of the rules governing ships at sea, there is no hope for success. These rules should be expressed at the right level of granularity for human comprehension, not at the level for system implementation. Consequently two cooperating groups should be started. One group would determine the rules while the otherougoup would implement a rule interpreter and rule language powerful enough to support the domain experts' rules. A testbed of example data together with desired conclusions is needed to develop and to evaluate the system. Rule based system support routinized decision making. Only for those situations where rules have already been specified could the system possibly succeed. In novel situations, a problem solving system is needed. When a situation could not be handled by rules, the

problem solving system would be entered. If a solution is found, a learning system could try to capture the solution in the form of rules to be added to the rule based system.

STAMMER

STAMMER[9] is a forward-chaining rule based system for identifying objects detected by sensors on board naval vessels and for interactively explaining the identification process. The system operates in a continuing environment where later data may supercede earlier data. The system is expected to make and to justify conclusions based on partial and perhaps erroneous information. With each conclusion, the system associates a confidence, which may vary with time.

At the conceptual level, the data base for STAMMER consists of a set of relations. Unlike most production systems, the problem of naval monitoring demands that the data base be periodically updated by messages (decoded into lists of assertions or relations) from external sources. The need to deal continually with new information led to a novel rule interpreter based on streams. This is an instance where the pecular nature of the C2 problem required an extension of current AI technology. Through this

technique all recomputations are avoided.

Another extension to existing AI technology was required to handle rules of the form:

(friend X Y) ==> (friend Y X) (.9)

where the (.9) indicates the confidence factor associated with a rule. To allow rules of this form a new algorithm for combining confidences was invented. This algorithm allows the evidence for a conclusion to form a graph. Previous algorithms required that the structure of the evidence form a tree.

STAMMER also permits non-monotonic logic[10]. In monotonic logic, any conclusion reached cannot be overturned. When the reasoning allows rules of the form

(unknown position X) => (not (instorm X))

then one needs to deal with non-monotonic logic, a topic newly interesting to AI researchers. This was handled by dynamically computing the confidence in an assertion, while maintaining the evidence structure statically.

The evidence structure was also used by the explanation system. This subsystem was itself written as a mini-production system. A form of English allowed was highly restricted by the system. With some experience one could trace the entire reasoning path that led to any conclusion, even ones which no longer held to be true.

Command and Control STAMMER proves that elementary platform identification can be done by an AI system and that the process of identification is sufficiently rational to be believed by people. Acquisition of rules was the most critical problem, a problem which was not satisfactorily resolved. A number of natural extensions to STAMMER can be envisioned. In order to project the effect of hypothetical actions or goals, an intelligent simulator is needed. This requires the representation of goals, spatial regions, and temporal events, all of which are understudy within the AI community. Another extension would allow the system to infer goals from actions. This again requires basic work on understanding goals as well as domain specific work involved in identify the goals of ships. Finally a third extension is needed to handle those situations for which rules are insufficient. To explain the unexpected or anomalous or contradictory events, a problem solving system is needed.

Problem Solving and Planning

The first AI systems, Logic Theorist[11] and GPS[12], were general problem solving systems. These systems had limited power due to the catastrophic search spaces they generated. Later problem solving systems, such as STRIPS[13], ABSTRIPS, WARPLAN[14], and NOAH[15], cut down on

the search space, but the success, at this point, is still minimal.

A problem solving system is given an initial state, a desired goal state, and a set of operators or productions as input and tries to find a sequence of operators that will transform the start state into the goal state. Consequently a problem solving system can be implemented as a backward chaining production system. There is a clear distinction between production systems and problem solving systems. When a production system fires a rule, the conclusion may be unneeded and the effort wasted. Conclusions from frivolous rule firings are left in the data base. When a problem solving system fires a rule, it moves to a new state. If this state is not on the path to a solution, then backtracking occurs and previous changes are undone.

Problem solving systems were first intended as intelligent planning guides for robots. This problem was idealized to the Monkey and Bananas problem. Given a room, a box, bananas hung from the ceiling, possible some obstacles, and a monkey, problem solving systems tried to formulate a plan so that the monkey could reach the bananas. By idealizing the problem researchers concerned with different task domains could communicate and measure their methods. This problem was sufficiently simple so all systems could solve the problem. The slightly more complicated domain of moving blocks from one state to

another was a focal point of research in the seventies.

The first techniques to be tried were those involving search. A number of search algorithms were invented, the major ones being depth first, breadth, graphical, and heuristic or best first. Of these methods only breadth first is guaranteed to succeed, but the computational requirements make this approach unsatisfactory is most cases. Searching will not solve blocks world problem in reasonable times. Minsky[16] proposed that problem solvers should apply "planning islands" to reduce the search space. A version of these planning islands was used in the ABSTRIPS system, with some success. More work on abstraction spaces is needed.

Two other approaches to the blocks world problem have been suggested. Sacerdoti constructs a "procedural net" which carries information about how to order actions. His system succeeds in many cases, but Sacerdoti himself doubts that it will solve all blocks worlds problems. As was soon discovered, goals often consist of achieving a conjunction of predicates, all at the same time, rather than one after another. Warren and Iate have a different approach, roughly corresponding to the idea of interlacing plans for solving each separate subgoal. This method is computationally expensive. Warren claims that his method is complete, i.e. will solve any solvable blocks world problem, but his implementation, he admits, does not support this assertion.

Although this work seems to be of the toy variety, in fact it is completely general. Warren exemplifies this by using his planning system, with different sets of data and rules, to solve i) the monkey and bananas problem, ii) machine code generation and iii) robot movement problems. In each case the state and goal state consists of a list of relations. Each operators adds and deletes relation provided that the preconditions (another list of relations) are satisfied. This framework is extremely general, but more knowledge is needed to adequately control the choice of operators.

At this time I will only mention another approach to problem solving, that founded upon predicate calculus. In this view the initial state and goal state are defined by a set of predicates. The particular way in which state changes are made is based on the principle of resolution[17,18]. The large search space problem is severe in this formalism. Nearly a dozen strategies for lessening the search have been investigated, but none has been successful.

Problem solving system have not worried about the problem of explaining the choices. In some sense, the ends justify the means. The systems all involve some aspect of blind searching. Currently the blocks world is the limit of their capability.

Command and Control Unlike production systems, no problem

solving system's ability is near that of any human. Their are a number of ways problem solvers might extend their power- by learning, by abstraction, by modeling, by generalization, and by analogy. Problem solving is necessary whenever the unusual or unexpected happens. This is precisely the time when humans need assistance. This assistance must be supported by reason and explanation. Another area of research to be supported should be problem solving explanation systems.

Natural Language Processing

The study of natural language processing began with the attempt to translate automatically from one language to another. The difficulty with natural language understanding was not appreciated at that time. What would a system that understood natural language have to do? Would it be sufficient to parse the sentence, i.e. identify the words of the sentence with various parts of speech?

Consider the following common sentences:

1. John gave Mary a headache. (Could she give it back?)

2. John gave Mary a book.

3. John gave Mary an engagement ring.

4. John gave Mary a bloody nose. (Could John give just a nose?)

5. John gave a good performance.

These sentences have very distinct meanings. The understanding of a sentence requires more than just a syntactic labelling of the words and phrases. One needs to know a large body of world knowledge, including such concepts as goals, causality, temporal relations, social relations, work ethics, ad infinitum. For many of these concepts, no satisfactory representation scheme exists. The area is still young and progress is being made.

Operationally a system understands a sentence or paragraph if it can:

1. paraphrase the statement

2. summarize the statement

3. expound (fill in details) on the statement

4. answer questions about the sentence

5. make inferences from the sentence.

The depth of understanding can be measured by the nature of inferences the system makes.

The meaning of a sentence is dependent on world knowledge and the current local context. Work in natural language processing can be divided into two major areas. One area is concerned with the representation of that body of knowledge which is necessary for language understanding. This area overlaps with the study of knowledge representation in general. The other area is concerned with providing a natural language interface to some existing data base. In this case the meaning of a sentence translates into some query or update of the data base. This is a much simpler problem than understanding natural language in general contexts. Rusty Bobrow has a parser which he believes can effectively interfaced to different domains within a three month period (he has done this several times). This system is not small and would require an additional 100k of storage.

Command and Control

The most comfortable interface to a system, especially for naive users, would be English. The cost, applying the RUS parser, would be several man-months plus an enlarged system. When the English front-end is absolutely needed, and the cost can be afforded, at that time it should be imported. Natural language processing for C2 poses no special problems. (I am not referring to processing coded messages, but to accessing a data base.) There is no need for the Navy to support or direct work in natural language

processing. Research in this area is needed and will continue.

Games

Game playing programs have always held a fascination for AI researchers. Games provide a microcosm to test ideas. The measure of performance is clear and the results easily understood. Chess[19] was the first game studied, and for two decades the variety of techniques yielded only very poor play. During this period a number of different search strategies were studied, including depth-first, breadth, alpha-beta, best-first, graph, and progressive deepening. The clearest result defines when search techniques will work.

The complexity of various games can be measured by the average number of legal moves one has per turn. For chess this number is about 35. For other games, such as checkers, backgammon and go, the average number of moves is 15, 800, and 200 respectively. One important measure for search algorithms, is the number of nodes generated in order to lookahead k moves (actually half-moves, called ply, is the usual parameter). The alpha-beta search will effectively search to depth k ply by considering only k*exp(2*square-root d) moves, where d is the average number of legal moves. Consequently searching techniques are successful in both chess and checkers. Currently chess-playing programs play at the level of state champions

and in blitz chess, will give grand-masters a battle. Checker playing programs[20] are at the world champion level.

The high number of legal moves in backgammon and go prevents the search approach from working. Nonetheless, Hans Berliner has written a very successful backgammon program based on a smooth polynomial evaluation function. This program beat the current world champion in a match where the computer had all the luck. In this method each legal move is evaluated by twenty or thirty features and that move with the greatest weighted sum is choosen. A number of approaches have been tried for the game of go, including search, polynomial evaluation, rule-based, and pattern-based. All of these techniques have been unsuccessful. Indeed, the best programs can only beat players who have played fewer than two games. Twenty years of research has not produced a competent go playing program. This is set forth as a warning to those who might blithely believe that programs can be easily written to perform at human levels of intelligence.

Search techniques do not afford a very deep explanation for any decision. A move is selected because it is best after considering millions of possible eventualities. The studies of deGroot indicate that in the game of chess, masters consider about 50 positions. Wilkins recently constructed a rule-base program which plays tactical

mid-game situations and generates only a few dozen positions. To do this a great deal of knowledge is applied and the explanations of the choice of moves are more comprehensible. The success of search in game-playing has been a disadvantage to the development of more comprehensible methods. The performance measure of "good play" is too restrictive. A program that plays well without the ability to explain its decisions has the characteristics of an idiot-savant. Approaches along the lines of Wilkins thesis will eventually lead to a better understanding of how people reason.

Command and Control To put command and control into a game-playing methodology requires a model which has legal board together with legal moves. The games that have been studied have perfect information (complete knowledge of your own forces as well as the opponents). The game of bridge affords a more realistic model. On defense one has a partial knowledge of your own forces and the opponent's forces. While the highest level goal is evident, the particular subgoals required to achieve those goals are manifold. Deception is an ever-present part of the game. To model this requires the ability to imagine how the opponents are viewing the situation. One need to draw inferences from what is not done as well as from what is done. As a profession, lawyers make the best bridge players. The game demands consistent reasoning. The post-mortem requires logical explanations. The persistent

presentation of reasons for bids and plays makes this game an ideal candidate for a prototype problem solving system with explanation. The novelty of situations demands more than a rule-based system and the unknow nature of the undisclosed hands prevents a search technique from succeeding.

Learning

When the study of AI first began, hopes ran unbridled. The difficulties of programming could be overcome by simply having programs which learned. The process of learning itself could be the subject of learning. Programs would not only get smarter, but they would learn at an constantly accelerating rate. Measured against these hopes, AI learning programs have been a dismal failure. However, let us examine the progress that has been made, measured against more realistic standards.

Samuel's checker playing program was the first successful learning program. It had two different learning components. One component adjusted the coefficients of its polynomial evaluation function in light of past experience. The other component acted as a rote memorizer to avoid previously done computation. With these techniques his program quickly learned to play very competent checkers. His methods tell us how to tune a system.

Buchanan, Mitchell and Feigenbaum developed META-DENDRAL [21,22]. This program inferred rules of chemical fragmentation and migration from input/output mass spectrometry data. The inferred rules were interesting enough to be published in Chemical Journals. META-DENDRAL is the only program to synthesis new knowledge. Other learning programs attempt to reinvent earlier discoveries.

Lenat's program AM [23] discovers relationships among natural numbers and conjectures properties of numbers. This system was a rule-based system where the rules were those that embodied some of the heuristics governing mathematical discovery. Through experimentation AM made more than 100 conjectures, including:

1. i) each number can be written uniquely as a product of primes

2. ii) each even can be written as the sum of two primes

3. iii) the sum of two even numbers is even.

Part of the process of discovery included discovering that the concepts of primeness, evenness, sum, and product would be useful. This program did not verify any conjecture nor did it discover any new fact about numbers. Rules capturing some aspects of aesthetics were part of the control of the system.

Recently Langley[24] has reported on a program, BACON, which infers physical laws from empirical, numeric data. BACON is a production system which has inferred such laws as Keplar's third law, inverse square law, Coulomb's law, Snell's law and others. Part of the discovery of a law is guessing that there is a relationship among the appropriate parameters. This is done by the human. The system tries to find a relationship among hand-picked data. No new law has been discovered, but unexplained data is being sought after.

Command and Control From the work that has taken place, an AI system to adjust coefficient or to build its own rules can be given a good chance of success. To infer rules, appropriate data must be generated, either by humans, as in BACON, or by experimentation, as in AM. The C2 context uses rules as part of a problem solving system. There are two possible avenues of research here. One would be to build a problem solving system together with a rule inference system. Rules could be inferred from traces of solutions of problems. Alternatively, given the input and desire output, rules could be generated following the methodology in META-DENDRAL.

Knowledge Representation

The concerns of knowledge representation will be introduced by analogy. Suppose we had a program which kept information on all the employees of some firm. We would want to able to add and delete information about employees.

Depending on the type of information kept, e.g. salary, address, health records, peer reviews, etc. we would require different data storing techniques. If we wanted to ask who made what salary as opposed to or in addition to what salary an individual made, this too would require a different data structure. This problem is not hard because we know what information should be stored and the techniques for updating and accessing the information.

Knowledge representation is the study of methods for storing that information necessary for human-like reasoning and inferencing. The updating of a knowledge representation is called knowledge acquisition. Knowledge application is the methods by which the knowledge representation is used. All of this topics fall under the heading of knowledge engineering. The three major knowledge representation schemes are rules, predicate calculus (or first order logic), and semantic nets. Much of the work in knowledge representation has been stimulated by natural language processing, for, as we have seen, inferencing is a necessary part of natural language understanding.

A recent SIGART[25] (Special Interest Group on Artificial Intelligence) was entirely devoted to Knowledge Representation. This article surveyed several hundred AI workers and obtained a consensus on the sixty topics asked about. Knowledge Representation is concerned with the computer representation of: goals, plans, causality, space,

time, process, contexts, abstration, generalization, specialization, plurality, quantification, prototypes, structure objects, individuals, attributes, ambiquity, descriptions, references, denotations, connotations, analogy, evalutation, simplificiation, induction, deduction, chunking, partial matching, beliefs, hypotheses, models, certainty, primitives, contradictions, consistency, completeness, adequacy, and a host of other topics. Of all the topics surveyed, only those dealing with the representation of logical issues, such as quantification, negation, conjunction, and disjunction were generally thought to be understood. Some areas, like the representation of spatial and temporal knowledge were thought to be understood by no one. Of course one could represent this information by means of coordinates, but is clearly not how humans do it, and does not provide any insight into how one would use the information to reason. This particular ignorance touches on the frame problem, which is the problem of representing information in such a way that only relevant updates are done, e.g. consider the amount of updating necessary to represent the changing perspective of a person walking through a room. Somehow people cope with this without becoming computationally swamped.

Command and Control A number of areas of knowledge representation are of particular concern to C2. These include the representation of goals, plans, space, time,

contexts, beliefs, and certainty. In this infancy period of Knowledge Engineering, two separate problems are often confused. One problem is the general problem of how to represent some quality or facet of knowledge while the other problem is the specific problem of representing the particular facet of knowledge for some domain. For example in the understanding of goals, knowledge representation techniques are needed to handle the general problems of competitive goals, of cooperative goals, of subsumption goals and the like. Also, for the application of these techniques, a specific domain analysis is required to identify the goals in a C2 environment. Domain analysis is not a simple task. Many people have tried to make explicit the rules for playing good chess without success. Recently Wilkins[26] has had some success at this domain analysis task. A necessary condition for successful domain analysis is a competent knowledge of the domain.

Conclusions

Specific recommendations for developing and using AI technology have been made throughout this paper. In general the Navy should proceed along two lines. By supporting basic research within the AI community in areas which are of particular value to C2, the Navy can attract the interest and labors of a greater number of people than those whom they support directly. As developed in this paper, areas requiring additional research are planning; representing, applying and identifying goals; and problem solving. These

areas can developed by allowing researchers to work in toy domains, domains which will have the same essential difficulties as C2, but which avoid the specific domain knowledge.

The second line should be the continuing development of a prototype system. One form of benchmark could be scenarios illustrating desired behavior. Another form would be a family of questions and question types which the system should be capable of answering. A third, and most desirable, form of benchmark would be the use of such a system in standard war games. The development of such a system hinges upon readily available data.

References

[1] Shortliffe, E.H., Computer-Based Medical Consultations:MYCIN, Elsevier, 1976.

[2] Davis, R., and King, J. An overview of production systems, Machine Intelligence 8, 300-332.

[3] Nilsson, N.J., Principle of Artificial Intelligence, Palo Alto: Tioga, 1980.

[4] Kibler, D.F, Neighbors, J.M, and Standish, T.A., Program Manipulation via an Efficient Production System, Proceedings of the Symposium on Artificial Intelligence and Programming Languages, August 1977, 163-173.

[5] Barstow, D. Knowledge-Based Program Construction, New York:North-Holland, 1979.

[6] Pople, H.E., Jr. The formation of composite hypotheses in diagnostic problem solving: an exercise in synthetic reasoning, Fifth Internation Joint Conference on Artificial Intelligence, 1977, 1030-1037.

[7] Feigenbaum, E.A., Buchanan, B. and Lederberg, J., Generality and problem solving: a case study using the DENDRAL program, Machine Intelligence 6, 1971.

[8] Doyle, J., A Truth Maintenance System, Artifical Intelligence 12, (1979), 231-272.

[9] Bechtel, R.J., Morris, P.H., and Kibler, D.K., Incremental Deduction in a Real-Time Environment, Canadian Society for Computation Studies of Intelligence,(1980), 26-33.

[10] McDermott, D. and J. Doyle, An Introduction to Non-monotonic Logic, Proceeding of the Sixth Internation Joint Conference on Artificial Intelligence (August 1979), 562-567.

[11] Newell, A., Shaw, J., and Simon, H., Empirical explorations of the logic theory machine, Proc. West. Joint Computer Conf., 1957, v1.15, 218-239.

[12] Newell, A., and Simon. H.A., GPS, a program that simulates human thought, Computers and Thought, 1963, 279-293.

[13] Fikes, R.E., and Nilsson, N.J., STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, 1971, 2(3/4), 189-208.

[14] Warren, D.H.D., WARPLAN: a System for Generating

Plans, Memo 76, Dept. of Computation Logic, Univ. of Edinburgh, Sochll of Artificial Intelligence, June 1977.

[15] Sacerdoti, E.D., A Structure for Plans and Behavior, New York: Elsevier, 1977.

[16] Minsky, M., A framework for Representing Knowledge, Psychology of Computer Vision, 1975, 211-277.

[17] Robinson, J.A., A machine-oriented logic based on the resolution principle, JACM, 1965, 12(1), 23-41.

[18] Kowalski, R., Algorithm = Logic + Control, Comm. ACM 22, 7 (July 1979), 424-436.

[19] Berliner, H.J., A chronology of computer chess and its literature, Artificial Intelligence, 1978, 10(2), 173-199.

[20] Samuel, A.L., Some studies in machine learning using the game of checkers, Computer and Thought, 1963, 71-105.

[21] Buchanan, B.G., and Feigenbaum, E.A., Dendral and Meta-Dendral:their applications dimension. Artificial Intelligence, 1978, 11(1,2), 5-24.

[22] Buchanan, B.G., and Mitchell, T.M., Model-directed learning of production rules., Pattern Directed Inference Systems, 1978.

[23] Lenat, D.B. AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search, Rep. STAN-CS-76-570, Stanford University, Computer Science Dept.; July 1976.

[24] Langley, P., Rediscovering Physics with Bacon.3, Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 1979, 505-507.

[25] Brachman, R.J., and Smith, B.C., Special Issue on Knowledge Representation, SIGART, Feb. 1980.

[26] Wilkins, D., Using plans in chess. Sixth International Joint Conference on Artificial Intelligence, 960-967.