MASSACHUSETTS INSTITUTE OF TECHNOLOGY
A.I. LABORATORY

Artificial Intelligence
Memo No. 269

October 1972

PROPOSAL TO ARPA FOR CONTINUED RESEARCH ON A.I.

Marvin Minsky

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0003.

PROPOSAL
to the
Advanced Research Projects Agency
for

Continued Research on Artificial Intelligence

at the
Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Principal Investigators: Marvin Minsky and Seymour Papert

Proposed Support Period: 1 November 1972 through 31 October 1973

Proposed Budget: $1,000,000

Approvals

Principal Investigator

Professor Marvin Minsky ______________________

Administrative Director,
M.I.T.'s Division of
Sponsored Research

George H. Dummer ______________________

Proposal to ARPA for Work on Artificial Intelligence
M.I.T. Artificial Intelligence Laboratory

Preface

The Artificial Intelligence Laboratory proposes to continue its work on a group of closely interconnected projects, all bearing on questions about how to make computers able to use more sophisticated kinds of knowledge to solve difficult problems. This proposal explains what we expect to come of this work, and why it seems to us the most profitable direction for research at this time.

The core of this proposal is about well-defined specific tasks such as extending the computer's ability to understand information presented as visual scenes--or in natural, human language. Although these specific goals are important enough in themselves, we see their pursuit also as tightly bound to the development of a general theory of the computations needed to produce intelligent processes. Obviously, a certain amount of theory is needed to achieve progress in this and we maintain that the steps toward a deep theory in this domain must include thorough analysis of very specific phenomena. Our confidence in this strategy is based both on past successes and on our current theory of knowledge structure. These bases are discussed both below and in the appendices.

To give a clearer idea of this position, we contrast our strategy with a number of questions typical of an earlier phase of thinking about Artificial Intelligence (and which are still often posed by outsiders):

> You seem to be making "performance" programs for certain particular jobs. Why do you concentrate on "specialist" programs like VISION and the BLOCKS WORLD, that deal with such narrowly restricted kinds of objects?
>
> Why don't you do more work on learning? Aren't you building too much of the solutions into your programs, instead of getting them to adapt themselves to new situations?
>
> Why don't you pursue more "general" problem solving methods? Are you using the right kinds of computers? Shouldn't you model the brain more closely?

Our answers to such issues are all tied closely together. Of course we want to develop programs with very broad powers. But finding how to do this involves the usual paradoxical strategy of Science: the keys to "generality" often lie in understanding thoroughly certain issues that are best studied in special situations in which the issues are particularly clear. We believe that for us the most central such issue is this: how can we exploit diverse kinds of knowledge in the same process?

> The art of computer programming has developed along lines very narrow in this regard. Each kind of "subroutine", in traditional programming, expects to get information in one specific format, "data type", or structure. Two processes operating at different "levels" then cannot communicate directly. The traditional solution is to organize all complicated programs into a series of stages or "passes" in which the data-types change sharply and irrevocably. Unfortunately, as we and others have found, this is incompatible with many important intelligence-like processes.

Our proposed solutions are still evolving, but they all seem to turn around new methods of programming and new ways to represent knowledge about programming. The field of Artificial Intelligence has made enormous progress in the past few years toward becoming a scientific subject. Among its most powerfully productive paradigms are:

Learning Structural Descriptions
Heterarchical Programming
Knowledge about Procedures
Procedural Embedding

In our proposals for the past two years, we explained why we see these as important and how we plan to develop them. We append the central portions to this proposal, as appendices. Readers who have followed AI closely should have no difficulty with the language of those expositions, but readers unfamiliar with the jargon of this field would do better to begin with our book-length report A.I. MEMO 252, a general introduction to the role of our work in relation to other sciences concerned with thinking and intelligence. The central part of this proposal will spell out the state of the particular projects to which those general ideas are applied in our Laboratory.

For some years, we have been developing these new methods largely in the context of two "micro-worlds", the VISION SYSTEM and the BLOCKS WORLD. Within these models we now can work on sophisticated techniques for learning, as well as on the new styles of programming. While both of these older worlds are very specialized in a number of ways, a third domain is now beginning to crystallize; it is a more general, common-sense domain of discourse in which one can discuss many everyday concepts we are sure must be involved in "thoughtful" activity--such matters as time, space, purpose, planning, simple economics, need for knowledge, etc.

GOALS and APPLICATIONS

The next few sections are about our main goals, both for long-range research and for particular application areas. The general technical position of our 1970 and 1971 proposals still represents the direction of our approach. However, although those proposals are relatively explicit about the high-level problems motivating our research, they do not give a very clear picture of the actual projects, or of their practical consequences. In this proposal we shall concentrate on giving a more concrete view of our immediate goals. The next four sections are each directed to an "area" of application. It will be obvious that the concepts and methods used in these areas are closely related. Our intentions about the applications vary, naturally. In some cases we have major efforts toward completing effective operational prototypes. In the rest, we expect to produce demonstration prototypes, or only to develop some theory, clarify problems, and attempt to direct the attention of others to problems which we think need immediate attention. Some projects are limited because we don't yet know how to proceed faster, others are limited simply by the size of the project staff, or by no-longer adequate hardware.

AREA I: ROBOTICS, VISION, AUTOMATION

AUTOMATION and PRODUCTIVITY:

Even in its current early stage of development, AI is ready for application in many practical areas such as industrial assembly, fabrication, inspection, mining, medicine, undersea and space exploration, and agriculture. Although we are not directly active in these areas, the AI Lab makes efforts to inform individuals and agencies who might be so involved. The United States has not addressed the problem of general productivity with the imagination and energy appropriate and necessary to meet future needs. The Laboratory has probably had a large effect in drawing attention to this situation and the past year has seen a substantial growth of concern with this area, in government and industry.

In conjunction with development of the mini-robot, we plan to work toward a demonstration of its applicability to automation. The target application is not yet settled; the front runner is automated assembly, inspection, and testing of very small computer processors, built of microelectronic components. This choice of application might seem a little "inbred"; the main factor is simply that the microelectronic industry is the only one really concerned today with very small assemblies. We are still open to suggestions about other applications.

ROBOTICS:

We would like to see much more sophisticated computer controlled effectors become available for practical application as well as for research. There has been much too little development in the field of mechanical manipulators. Even though there are perhaps 50 competing "industrial robots" in production today, they are all astonishingly uniform in their primitive articulation. None have but the most rudimentary feedback facilities. Although it would be a serious diversion to try to make our Laboratory into a center for general development of reliable mechanical hardware, we and the other ARPA projects are playing a critical role in drafting prototypes and stimulating national activity in that area.

In particular, D. Silver is now debugging a force-feedback system that uses strain measurements in the manipulator's wrist. This appears to be extremely effective for practical matters, as a compromise between elaborate touch sensors (that are hard to relate to larger-scale forces) and motor-load feedback systems (which in principle inform about local forces but are impractical in practice because of the very poor signal/noise ratio one has after subtracting out gravity and inertial terms for the whole moving mass).

COMPUTER VISION:

Our long-range goal is to discover how to make machines able to look out at the world and "see". This plays such a large role in our previous work that we will not review it at this point, except to note that this goal involves major developments in areas often called "perception" and "pattern recognition". We maintain that good performance in non-trivial vision problems requires systems that really understand a lot about what they are looking at, and the kind of fundamental research on knowledge that we are doing can not be side-stepped.

The MIT VISION SYSTEM is, at present, still an experimental prototype developed to explore new ideas both about machine vision and new styles of programming. It is the most powerful system available within its narrow domain of analysing monocular, polyhedral scenes. We are extending its capabilities to deal with more natural objects. Up to now, this system could deal only with polyhedral objects having uniform plane surfaces, such as blocks, wedges, pyramids and the like. But

(1) We are now working on processing and description of textured surfaces.
(2) More global object-descriptions are being studied, using a scheme of Hollerbach that starts with basic shapes and describes local deviations from it; we expect to be able to extend these to non-planar objects.
(3) The new system will be able to deal with motion and use it for

visual purposes in applications such as pouring liquids.

(4) We are beginning to combine visual, tactile and force feedback, and plan to demonstrate a significant automatic assembly application, using the new mini-robot hardware.

We hope to show that the sophistication developed over the years in our heterarchical vision system will lead to a system able to deal with seeing real situations like that in a regular tool box. Although the problems in a realistic environment will be more complex, we expect many principles to carry over from our present system. Among these principles:

The best methods for "low level" scene analysis lean toward relatively simple low level filters guided by large amounts of knowledge about the objects.

Great power derives from the use of a firm theory of the semantics of boundaries to rapidly classify lines as shadows, cracks, concave or convex edges, etc. Better results come from using higher level knowledge than from attempting to push the use of local optical cues beyond meaningful limits.

We know a good deal about building descriptions in terms of geometric and optical relations between objects, ways to hypothesize objects from partial presentations and ways to propose groupings and using the groups to aid analysis. It remains to be seen how smoothly this experience can be transferred from the old BLOCKS WORLD to the realistic scene world.

MINI-ROBOTICS:

We have a particular interest in developing laboratory equipment for robotics research on a very small physical scale. We have two reasons for this. First, there are important applications for a mechanical "microtechnology" that does not now exist, yet there is no serious technical limitation deterring it. These applications range from involving fabrication and assembly of physically small but complicated systems to new applications in medicine, agriculture, and space that are generally unrecognized at present. For details, see our proposal to ARPA about Microtechnology. Second, we feel that there is now a sturdy backlog of good ideas

> for advanced automation that are developing very slowly only because it is hard for people to set up research laboratories in this field. This project is an opportunity for our Laboratory to simplify and modernize its own equipment and, at the same time, produce an easily-copied system that we hope will have a substantial effect in speeding up progress in other centers.

We emphasize that the size scale involved in our mini-robot project is NOT so small as to require major engineering innovations. The devices will work within a space of a few inches, with accuracy on the order of .001 inch, which is a modest precision in machine tool practice. We expect the system to be able to handle tasks on an order of magnitude smaller than does current equipment, without creating very serious new design complications.

AREA II: Knowledge

"The trouble with computers is ...!"

Everyone has his own complaint, deriving from some collection of bad experiences. The computer is too literal; you have to tell it about every possible situation, and even then it will manage to find some misinterpretation. Another way to put it is to say that today's computers have no "common sense". For example, in the BLOCKS WORLD, we have to write explicitly into the program something that says not to try to put two blocks in the same place. Less trivial:

> If there is a statement that "block 1 is on block 2", and some action moves block 1, then that statement should be removed.

To do this explicitly for all kinds of statements leads to serious problems, because some such statements concern others. We need a more general method. Not too general, of course; if an action causes block 1 to be painted red, we don't want the action of moving it to make the machine think it will become green again! Solution: the machine should know that "geometric" aspects like "above", "parallel to", "to the left of" transform in certain ways, while "surface" properties like "green", "rough", "sticky", transform in other ways. These are things that "every child knows".

Even in situations that seem very simple to people, behaving in a "sensible" fashion requires knowing a lot about many different kinds of things. In one way or another, every facet of our Laboratory is concerned with such issues, as are the other AI laboratories. Ordinarily, in each particular problem area, one accumulates more and more "ad hoc" techniques that improve competence. Unfortunately, this kind of experience doesn't seem to lead to general insights. Formulating the theoretical issues is critical if we are to make sense of the whole area and discover practical methods for handling real world problems. The issues can be formulated along a number of dimensions:

BROAD vs. DEEP:

Most workers seem to believe that common sense reasoning is fairly "shallow" in the sense that if the process were represented as a logical deduction process the "proofs" would not be very long. Even so, finding a proof with just 10 steps would pose a terrible search problem, because it is clear that the basis for common sense reasoning must contain many thousands of items. The problem, then, is more one of finding a mechanism for determining "relevance" than finding one for deduction. Such a mechanism would depend, presumably, on building a classificatory structure for how different kinds of knowledge interact.

INDEPENDENCE:

How does one add new information to the system? In one ideal, all information is expressed in the form of compact "declarative" statements: "broken glass is sharp"; "powers of a group element form cyclic subgroups", etc. In the "theorem-proving" approach to reasoning, all the information is so represented on the same level, as "axioms", and new information is added simply by adding another axiom.

This ideal leads, unfortunately, to some serious difficulties. One really cannot use "facts" unless they are accompanied by information about how to use them. Some facts serve to warn against using other facts for certain jobs. A person's most important kind of information, perhaps, is about thinking itself; how to break a problem into subgoals, deciding when certain kinds of analogies are appropriate, when certain kinds of plans are realistic, etc. We don't know enough, yet, about how to represent information about such matters, independent of particular application areas, and this makes problems for the theorem-proving approach.

PROCEDURAL vs. DECLARATIVE:

These problems seem at first not quite so difficult when one considers embedding knowledge in the form of procedures. A warning about a certain "fact" becoming undependable in some kind of situation can be embodied by programming an appropriate test at the right place in the problem-solving program. But this only raises other, equally hard problems. Embedded in a procedure, the meaning of an assertion can become dependent on its local context to such a degree that inserting more knowledge becomes very hard. In the declarative system this problem of local dependence is not so acute, but it is only an illusion to think that it can be made to disappear effectively. For even in the axiomatic systems, the meaning of a term depends on other assertions using that symbol, and the interactions can become badly dispersed; one suddenly sees the local dependence as a lost advantage instead of as a nuisance.

An intermediate kind of system--the "production" of our colleagues at CMU--has many problems of its own; the effects of items are influenced largely by their neighbors, they have some degree of independence, but it is hard to see how they can apply to one another in ways that would be useful for planning.

This is an area of intense intellectual activity today, with very diverse viewpoints. Our plan is not to attempt yet to decide which approach is best, for each has important elements, but to attempt to extend our microworlds in several different ways to find what are the important problems and phenomena.

NATURAL LANGUAGE:

Work is continuing on various extensions of SHRDLU, Winograd's language understanding system. The availability of this system over the ARPA network, in a well documented version at CMU, will permit faster progress in developing techniques for introducing new concepts. Unfortunately, our own computer system is too overloaded to make the facility available to many outsiders, and it will no doubt load the other systems rather heavily, as well.

Work on language will continue to develop both theory and extensions of current capabilities. A variety of seminars are directed toward a deeper understanding of the processes involved in language understanding, with one goal--that of defining the directions for a "next generation" language understanding program. A number of students are working on aspects of syntax and semantics, such as the problems of quantifier scope, the addition of a wider range of syntactic constructions to the previously developed grammar, the issues raised by simple sorts of language translation and the problems of generating complex syntactic structures from a semantic base. Drew McDermott (see below) is writing a program which will make the sort of plausible inferences necessary for understanding language in context. This is being done in conjunction with the development of the CONNIVER language as a formalism for representing and manipulating knowledge. His program follows a narrative of a simple series of events, trying to draw causal connections between them and making predictions about what it expects to happen.

Several things are being done to make the earlier work (see last year's progress report) more useful to other people. Some of them are being done in conjunction with the Project MAC automatic programming group and the AI project at Carnegie-Mellon. They involve providing easier ways to interface programs with new data areas, a user-engineered front end enabling people to run the programs and interrogate them through the use of a simple command tree, improved documentation and other descriptive material such as flow charts of the grammar.

UNDERSTANDING NARRATIVE:

In his recent thesis, Eugene Charniak proposes a technique for handling many aspects of common sense in connection with understanding narrative. A narrative is a sequence of statements describing a sequence of events, understandable by a person, but which usually contains a near minimum of detail. A great deal of material must be filled in by the reader, using his own store of general knowledge, to convert a narrative to a step-like chain of statements that say everything explicitly. Charniak sketches out a sequence of stages of analysis for each sentence that may result in filling in many such details. These mechanisms are supposed to help resolve ambiguities, determine unspecified references, propose motivations for actions and generally act as information retrieval and plausible explanation devices.

Charniak's central mechanisms involve setting up procedures called "demons". These are essentially PLANNER antecedent statements that are set up by local contexts and then persist, actively "watching" for statements that might signify the occurrence of certain activities that particularly concern the demon. For example, when a mathematical subject is mentioned, the system could set up a demon that monitors the subsequent text for ordinary words that happen to have special mathematical significance, so that meanings would be considered that would have a very low rate of occurrence in ordinary text.

Last year's proposal mentioned plans for this system. It has changed and developed substantially and the thesis will be available soon as a report. There does not yet exist an operational system for exploring the advantages and limitations of this idea and constructing one will depend on recruiting capable workers for that project.

UNDERSTANDING NARRATIVE, 2:

While Charniak's work applies to a world that requires a very wide knowledge base--it is the world of things a first-grade child must know to understand his reading books, Drew McDermott is attempting a system that attacks the same general kind of problem in a less local way, by attacking a problem of narrower scope but greater logical depth. Charniak's system can be viewed as an experiment to see how far one can go in filling in narrative details by setting up predictions whenever possible to embody what one already knows or might reasonably expect about the situation. These expectations operate on a very wide range of subject matter but they have no general mechanisms for dealing with complicated interactions. McDermott's model operates in a much more restricted domain--an extension of Winograd's BLOCKS WORLD, which now contains an actor with his own motivations and knowledge. Again, the system has to generate a hypothetical scenario to be consistent with a narrative. The system must deal with different kinds of causality, decide which kinds of explanations would be adequate for an actor who has specific items of knowledge and generally have theories about what kinds of assertions are "plausible" under different belief conditions. In particular, the system must distinguish between actions that are plausible to it and those that are plausible to the actor who may not know that a certain object is behind another, etc.

INFORMATION RETRIEVAL:

This field, which addresses a problem of recognized and ever-growing importance, has been dominated up to now by marginally useful attempts to classify knowledge without understanding it. In any particular application, such an "information retrieval" system can be made to do a certain amount of useful screening but as the data base expands past the experimental prototype, one always finds a threshold that if set too low inundates the user with too much but if set higher misses too many relevant items. Obviously, one can escape this only by increasing the degree to which the system itself can deal with the meanings of the classificatory concepts. As our work on language-understanding progresses, we can expect more and more areas of application for such techniques. Our new "micro-worlds" of common-sense reasoning should begin to open up the area of information retrieval to effective applications of Artificial Intelligence research.

We do not have a committment to any specific project in this information retrieval area. However, we are convinced that it is approaching the time when such a study will be worth while and we plan to discuss it with others versed in bibliographic and library sciences.

AREA III: PROGRAMMING

Workers in Artificial Intelligence have played important roles in the development of many new kinds of programming languages. As far as our own purposes were concerned, the language situation in our Laboratory was, if anything, unusually stable over a decade; almost all AI work (except, notably, the Greenblatt Chess program) used LISP as its basic language, and LISP development itself was generally conservative. However, during this period several sub-languages were developed within the specialist program projects; for example METEOR and CONVERT were completed and used in cases where LISP seemed inadequate. These were early pattern directed systems related to COMIT and SNOBOL and, more recently, such ideas and many others were put together in Hewitt's PLANNER. While the earlier steps in this direction were exploited only by their inventors, PLANNER's time was evidently ripe, for it was quickly adopted for use by Winston's Vision project, Winograd's Natural Language project and several others in our Laboratory. Thus we have seen the first major language change in a long time. Many other AI centers are already working with versions of MICRO-PLANNER, an implementation of a subset of Hewitt's proposed large system. Now under development, jointly with Project MAC, is a more comprehensive system for PLANNER. In the meantime, experience with MICRO-PLANNER led to a reaction by some users who had difficulties with debugging in situation where PLANNER's unprecedented facilities for automatic search, backup, and revision of its own data-structures could get out of hand and also in situations where programmers wanted their programs to have access to information about just what happened within the automatic searches. This led to the proposal of CONNIVER, by Sussman and others, a language in which one has many of PLANNER'S specification and automatic facilities but with more control over multiple functional environments, return to previous states and explicit control of backup situations. I may take quite some time to settle the batch of new practical and theoretical arguments about which assortment of structures and features should be given priority in the "general purpose" AI language of the near future; in the meantime one has little choice but to encourage experimentation in this area. We list below activities bearing on this subject:

LANGUAGE DEVELOPMENT: PLANNER

Jointly with Project MAC, a PLANNER development project is under way. There are quite a few interesting problems of how to implement the new kinds of objects and quantities. It is not an ordinary systems programming problem. There are serious theoretical problems about the extent to which such programs can b compiled, and little is known about what running speeds are possible in any case.

LANGUAGE DEVELOPMENT: CONNIVER

Conniver permits direct cross communication between different past states of a running program. Thus, one can make a suggestion and ask how it would have helped in a previous attempt to solve a problem! At first sight, the implementation problems appear enormous but many students are thinking about this and we are optimistic that a system will be built that can efficiently run larger-than-"toy" problems.

LANGUAGE DEVELOPMENT: LOGO

The language LOGO, developed in our Education project, now has an incarnation within LISP, developed by I. Goldstein. This brings a useful tool into the ARPANET community of AI workers, because of the rapidly growing body of work in children's cognitive development that is involved with the use of this language.

LANGUAGE DEVELOPMENT: LISP

In association with MATHLAB, our version of LISP, MACLISP, has been under continual development, primarily by Jon White and others, and there is now a version accessible to the ARPA network.

PROGRAM-UNDERSTANDING PROGRAMS:

Elsewhere in this proposal, we mention several projects that are connected with language development. Although not directly concerned with new major languages, each has a specialist language of its own and some of the ideas in them are sure to be demanded and implemented by others in one or more of the major exterior languages. Particularly promising in this regard are the projects that are developing programs which understand other programs, notably the experiments of Goldstein, Sussman, McDermott and Hewitt. The control structure of Winston's VISION SYSTEM, currently written in PLANNER, is also likely to stimulate new styles of procedure description.

THEORY OF PROGRAMMING:

The AI LAB has played a key role in persuading the computer science community to consider more realistic problems; to shift their attention from syntactic form to semantic content. In programming languages this means a move from computer-linguistics to theory of control structures. In mathematical research this means shifting from infinitary logic and recursive function theory to theories of computational complexity, and theory of solvability of control for simple problems. Both of these are just as "abstract" and theoretical, but far more realistic and practical.

Indeed, such results from recursion theory as "the unsolvability of the halting problem" can be deeply misleading. For an "interesting" class of practical programs, knowledge of the programmer's intent, and understanding of common control structures is sufficient to debug a non-halting program. Similarly, in the areas of "pattern-recognition" and perceptual sciences, we have helped to turn the emphasis from "invariants" to goal-relevant descriptions and, generally, from a wishful-thinking search for the Gestalt to the careful analysis of ways in which processes can pass from local features to global descriptions and back, in heterarchical control structures.

While we plan to continue working in these areas, it is impossible to predict which will develop next or in which way. While we stand by our judgements about priorities of importance in this sort of theoretical research, our Laboratory is in danger of becoming ineffective in implementing our opinions, for budgetary reasons. We cannot afford to recruit enough staff to maintain first-rate competence in the necessary areas. We plan to look for additional support in the theoretical area.

AUTOMATIC PROGRAMMING:

Many of our staff members are working on projects that bear on a common problem; how to create a computer program that satisfies a collection of goal-oriented conditions--that is, a program that accomplishes a task which is specified in terms of the result rather than in terms of the computational steps to be taken. This is true "automatic programming" and the possibility of doing it resides essentially in the degree of our ability to represent within the computer appropriate kinds of knowledge about computation, programming, debugging, planning, etc. As the body of this proposal shows, our main current goal is to develop this kind of knowledge and techniques for representing it and we expect that this body of work will lead, in the not too distant future, to real progress toward programming systems that can be told what kinds of programs are to be produced.

AREA IV: Education and Prosthetics (Non-ARPA projects)

EDUCATION:

To give a rounded picture of the Laboratory's goals, we have to mention some work not supported under the ARPA contract, but mutually interdependent with it. The largest such activity is a program of research into the structure and development of human cognitive skills, with emphasis on developing ways to help people learn better how to learn. In this area, our Laboratory is working on theories originating jointly in Artificial Intelligence and in the kind of developmental schemata proposed by Piaget and his colleagues. We feel that the discoveries we have made in this area have been of major importance both in formulating new ideas about design of superior educational paradigms, and in helping us formulate conjectures about how common sense knowledge may be structured for use in an artificial intelligence.

PROSTHETICS and MOTOR CONTROL:

A natural result of combining ideas about artificial intelligence, robotics, and human problem-solving is a new horizon of possible ways for men to control external devices. In instructing a robot to do a task, the ordinary industrial technique of recording the precise trajectory of a master-slave manipulation system will work only in the rigid assembly-line situation of absolute task repetition. An intelligent helper, on the other hand, could do much more with an explanation or some meaningful gestures. We are planning some studies in which highly handicapped persons might control external systems, such as virtual keyboards or simple but versatile manipulators, by using limited motor channels augmented by sophisticated intention-guessing software. One immediate goal may be to establish useful channels for intelligent but isolated cerebral palsy victims to establish effective real world interactions.

Technically, this project resembles the ARPA speech program, in the way that limited objectives can have large values in the right situations. The scientific details may also turn out to be rather similar. We plan to approach a Health agency for support in this area, and to have this activity involve people both in our Robotics and Education groups.

PART II: ---------------- General Theory of Intelligence

Why do we believe that we can achieve these goals?

The complexity of the answer depends on the degree of achievement one has in mind. The more immediate applications are easily seen to be "within the state of the art" set by recent work in our Laboratory and elsewhere. This includes the construction of a working mini-robot, extending machine vision to some industrial situations, and so on. Other aspects of our goals cannot so easily be justified. Their achievement would depend on solving hard theoretical problems. Our confidence that this can be done rests on an elaborate theory of intelligence that reflects many years of work. In this section we review some of the issues in constructing this theory of intelligence. It is not fully self-contained here and must be read in conjunction wit our progress reports, especially A.I. Memo 252 which is appended, and our previous proposals.

> The form of our review is a brief, almost telegraphic, statement of the lines of approach to the problem of intelligence most common among computer scientists. Its purpose is to situate our own position in an intellectual landscape that will be familiar to the reader. That is why we will not review the many other, quite different opinions

held in other disciplines.

ADMINISTRATIVE POWER:

The secret of intelligence lies in a problem-solving structure that is inherently highly efficient and selective, by virtue of some particularly good schemes for deduction, bookkeeping, goal-tree administration, redundancy-elimination, etc.

LOGICAL UNIFORMITY:

The secret of intelligence lies in using a universal logical decision procedure. This should be able to handle all sorts of problems and not need special programming for each new kind of problem. Of course, one must be able to suitably encode the different problems into it along with an adequate set of efficiency-enhancing heuristics.

COMMON SENSE:

One can't expect much "general intelligence" unless the system knows the simplest kinds of things that every normal child does; if you move something it will no longer be where it was; two things can't be in the same place; if A precedes B and B precedes C then A must precede C; etc. Therefore we first should concentrate on a program with "common sense". This specialist program is essential for other developments.

EPISTEMOLOGICAL POWER:

A person seems intelligent because he knows how to deal with a large variety of problems. We therefore have to make collections of such knowledge and to do this we first have to develop systematic schemata for acquiring, representing and classifying all the different kinds a knowledge that will be needed.

LEARNING:

It is impractical to do all the work implied by the previous two suggestions. It would be better to make a general learning program that will acquire both common sense and general knowledge by trial and error, reading books, etc.

PROGRAMMING KNOWLEDGE:

Perhaps the secret of the general problem solving ability of intelligence lies not in having enough adequate procedures in advance but in being able to assemble, quickly, a problem solving program appropriate to the problem of the moment! This would still mean that one needs knowledge, but now it isn't so general; what one needs is to know enourh about programming (and heuristic programming, in particular) so that one can write and debug a program that is good enough for the task.

Before proceeding further, we should summarize our attitude toward these viewpoints. There may or may not turn out to be other unsuspected general principles of great importance. We take the position that all the above points are relevant but that one must also add:

"GENERAL KNOWLEDGE":

What we see as "intelligence" is a system's ability to solve diverse and difficult problems, in impressively short and effective ways. When a system solves a problem in a manner that does not use tedious search, we must say that it seems to know how to solve the problem. For various technical reasons, we cannot believe that there can exist "general" methods to do this on a wide variety of problems without a lot of stored knowledge, so a lot of intelligence, on the performance level, must be in knowing how to solve problems efficiently--and specifically.

KNOWLEDGE ABOUT KNOWLEDGE:

On the other hand, for other technical reasons we think that simply holding large stores of knowledge is inadequate, because performance will degrade--unless much of the knowledge is structured to direct the use of the rest. Much of what one knows, then, will not appear at the "factual" level at all, but will concern when and how to use other parts of the information store and, especially, when not to.

Although a fixed store of such knowledge could be adequate for a vast variety of useful activity, some of it should be concerned with how to acquire more, so that intellectual growth is possible. How much knowledge-structure is needed to begin the "bootstrap" process? We don't really know much about this. Our conviction is that one must be very careful in theorizing about the development of intelligence in humans, not to overlook the amounts that may be absorbed from the environment in highly-prepared abstract form, in the course of acquiring language. Putting aside questions about how the linguistic processes themselves are developed, we believe that even at the level of learning proper use of individual words, the child is more or less forced to acquire thousands of important concepts whose importance has been determined by millenia of forceful selection.

We have not mentioned many other opinions held by computer scientists; for example, that perhaps the important advances lie in parallel processing, or in holistic access to content-addressed memory, etc. etc. We have noted only some of the views that play significant roles in our own models of the situation.

Even without attempting to decide between these--or rather, discussing the roles that they each play--it should be clear that such a discussion presupposes a thorough understanding of the issues. Exactly here lies the importance of the relatively deep, if narrow, examinations of the "specialist" programs. Both the VISION SYSTEM and the BLOCKS WORLD have told us a great deal, not just about how "performance-level" knowledge might be structured but also about how problem solvers must be programmed. The new wave of programming methods--PLANNER, CONNIVER--and the Heterarchical approach in general, are all attempts to meet problems that arose in these studies; issues that never clearly crystallized in the earlier studies on more "general" problem solving systems.

LEARNING

Here again there are a number of positions that each show only part of the situation. One common view is, we feel, really wrong; that there is a process called learning which is quite distinct from other mental activities. We list a number of principles that may seem obvious to some readers, and perhaps wrong to some others.

DESCRIPTION:
To record the important consequences of an experience, one must extract and represent the essential features. So learning is not merely recording; it involves shrewd abstraction. There is little use in recording everything.

UNDERSTANDING THE GOAL:
We ought to know the kind of program structure we want to result, before we can expect to be able to design a program to learn that structure. If learning then consists of adding axioms to a logic data base, the learner has to understand something about the "proof" procedure. If learning consists of making changes in programs, then the learner must know appropriate things about programming.

INCREMENTALITY and CONTEXT-INDEPENDENCE:
We want to be able to make small changes easily. Adding something new shouldn't distort everything learned before. So far as possible, the learning mechanism shouldn't have to take into account everything else already known; nor should it have to know all about how everything is organized. On the surface, at least, this would seem to favor use of declarative statements, or production-based programs. But we do not really understand the extent to which these make it possible also to represent control structures in ways that are also easily changed incrementally, and we suspect that declarative structures will not turn out to be very powerful in this regard.

DEBUGGING:
Some learning, at least, is rearrangement and debugging, not just adding new performance-level knowledge, but reclassifying it,

restructuring it, modifying the procedures that process the data structures in which the "factual" knowedge is embedded.

LEARNING TO LEARN:

We can't really expect, after all, to build a program that can do all these things from the start. So the learning program itself had better be self-applied so that it can be bootstrapped--by a good teacher.

Our experimental and theoretical program must recognize that because we do not yet have sufficiently intelligent paradigm programs, we simply cannot yet feel sure just what sorts of structures we will want our learning programs to learn! That is, although we cannot see any clear and particular difficulty ahead in building learning programs, we do not yet know enough to "specify" them. At the risk of seeming to repeat, it can be put this way: we still do not have any prototype programs that exhibit satisfactory "common-sense" behavior using shallow reasoning and large knowledge bases. Not having such a model for a target, we are not quite ready to attack directly the problem of learning such a structure and still less ready for the problem of "learning to learn" it.

The most promising system for learning at a symbolic description level is, at present, the one presented in Winston's thesis. That system is a specialist program; it works at building structural descriptions in the BLOCKS WORLD by "learning from examples". The basic strategy in this scheme is to compare a description of the current situation with certain "learned descriptions" that it has built up from earlier experiences. The "most important" differences between the present reality and the stored paradigms are computed by a procedure that embodies the machine's prejudices about what features of differences between situations are the most important. This general scheme raises a lot of specific problems for study:

Understanding differences.

The program of Winston's thesis builds and modifies descriptions by selecting one "dominant" feature from many, in each cycle of comparing two structural descriptions. This is done by a common-sense ranking of some kinds of relations over others. A more intelligent way would be to understand how such features are really related to the current goal; then the system could have "reasons" for its priorities. This would become important when, in new situations, it would be able to re-order these priorities in a rational way, by reasoning about its reasons.

Procedural learning.

With all that we have said about the importance of learning and debugging procedures, it would seem natural to ask about extending or adapting the structural description-learning program to program learning. At first glance, there seems a world of difference between describing stacks of blocks and describing computational schemata. However, Winston's chapter on "Grouping" suggests that these might not be so far apart. We see a possible bridge in the transformational process that recognizes repeated chains of structural relations and represents them by a non-repetitive structure describing the typical neighborhood of a "typical" element.

Such a "group" structure resembles the inner loop of a program. The abstraction process itself seems close to that one would use to generate a closed subroutine to compress parts of an iterative performance protocol (as proposed, for example, in Hewitt 1972). To make it write an actual program one would need, in principle, only to make it add to its representation of "typical element" information about the initial and final elements as well. For example, the program now describes a "stack" by describing a "typical element" and noting that each typical element has another typical element above it. To convert this to, say, a program for building stacks, one would want to add an initialization section:

"put a block on the table"

and an exit condition:

"stop if that block is two feet above the table".

The "inner loop" ("put a block on that block") would be generated by a procedure that understands how to make an element "typical", namely, by the action "put a block on it"!

PROGRAMMING EPISTEMOLOGY:
If we do that, have we made a large step toward understanding enough about program-learning, or is it a miniscule part of the job? To deal with such a question, one would have to work with a classification of program-structures, and at least a tentative measure of the importance of the different varieties. Clearly, one can do anything, in principle, with the initialize-loop-exit programming element, since the exit condition allows the expression of conditionals. Equally clearly, the loop structure, in practice, is a very confining, inexpressive structure for the kinds of procedures an intelligent system would seem to need.

PROGRAMMING and PLANNING:
The issue of how to represent abstractly, the protocol of an experience, is complicated by how representations interact with various kinds of goals. For example, one might ask for proposals about how a machine might have learned the processes that are preprogrammed into the problem-solving substructure of Winograd's BLOCKS WORLD.

Consider just the part of the system that removes all blocks on top of a block that the program wants to move.

1. One might have a process in which the machine solves a particular case and then generalizes. If several blocks are removed, and the protocol of actions is analysed, it would not be hard to detect the repetition and represent it as a loop.

2. In PLANNER code, however, there is no loop. One might instead recognize in the protocol the goal-subgoal-action relation of removing the supported block, and embed this in an antecedent theorem. Here, the generalization is easy, even from a single instance; what is harder is to see how to notice the significance of the repetition; in other cases the one-example generalization will usually be wrong!

3. In any case, the problem remains of generating a procedure appropriate to the goals. The CLEARTOP program is appropriate only under certain conditions. If there are a great many blocks on the one we want to move, then CLEARTOP will involve a lot of work. The system should be able to learn that this may be a condition for backing up to another goal; it should perhaps be able to use knowledge about sweeping all the blocks off with a broom; it should perhaps know about lifting the lower block and dumping them off (and this may necessitate some common sense about the randomness of where they may land).

4. We might really want to know how many blocks are on the one to be moved, so this kind of observation might have to be programmed in. What kind of knowledge is required to judge when counting is appropriate?

For some time, Gerald Sussman has been attempting to formulate the issues involved in different ways to look at the problem. Recently, he has constructed a program which "learns" programs like the CLEARTOP procedure by running simpler programs on examples, detecting "bugs"--incidents in which the program does something silly--and formulating a program change that makes it work on that example. It remains to be seen whether this approach to "learning through debugging" can be made powerful by providing it with enough diagnostic description and programming knowledge. It is interesting how issues arise in such a project that resemble problems in quite diverse areas of computer science; issues about multiple processes, management of temporary storage, domains of protection (of the results of interacting subgoals), etc.

PROGRAMS and INTENTIONS:

A direct attack on the problem of representing knowledge about programming is under way in the work of Ira Goldstein, who is developing a system that understands some structures of programs that produce graphical drawings. This project is interesting because it combines general knowledge about programs--loop structures, recursions, etc. with a semantic basis in geometry--an area that we have always found particularly rewarding because of the deep mathematical techniques that are close to the surface of such activities. Goldstein's program attempts to relate the expressed intention of the programmer--for example, a child who has written a bugged LOGO program intended to draw a figure of some specified kind--to the code he has written. It looks for clues about various kinds of familiar programming structures and should be able to debug the programmer's code even when the actual code is hopelessly flawed. Such a program must handle a variety of different kinds of knowledge:

about procedures

about geometry

about pictures and must explore how declarative descriptions of scenes relate to procedural drawing programs.