DOCUMENT RESUME

ED 152 294 IR 005 775

AUTHOR Kearsley, Greg P.
TITLE The Relevance of AI Research to CAI.
INSTITUTION Alberta Univ., Edmonton. Div. of Educational Research Services.
REPORT NO DERS-06-040; RIR-77-2
PUB DATE Feb 77
NOTE 25p.; Best copy available

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage.
DESCRIPTORS *Artificial Intelligence; *Computational Linguistics; *Computer Assisted Instruction; Computer Programs; Information Processing; *Man Machine Systems; On Line Systems; *Programing Languages; Research Needs; Thought Processes

ABSTRACT

This article provides a tutorial introduction to Artificial Intelligence (AI) research for those involved in Computer Assisted Instruction (CAI). The general theme is that much of the current work in AI, particularly in the areas of natural language understanding systems, rule induction, programming languages, and socratic systems, has important applications to CAI. A recommendation for more planned interaction between AI and CAI includes possibilities for joint conferences and joint research projects. (Author/VT)

February, 1977

DERS 06-040

RIR-77-2

THE RELEVANCE OF AI RESEARCH TO CAI

Greg P. Kearsley

The Division of Educational Research Services
Faculty of Education, The University of Alberta
Edmonton, Alberta, Canada
T6G 2G5

ABSTRACT

This article provides a tutorial introduction to Artificial Intelligence (AI) research for those involved in Computer Assisted Instruction (CAI). The general theme espoused is that much of the current work in AI, particularly in the areas of natural language understanding systems, rule induction, programming languages, and socratic systems, has important applications to CAI. It is hoped that this tutorial will stimulate or catalyze more intensive interaction between AI and CAI.

Artificial Intelligence (AI) means different things to different people. To the public, it is typically associated with chess-playing programs and building robots. Indeed to certain AI researchers, AI is the construction of game-playing programs or robotics. To others, AI is principally concerned with "studying the stucture of information and the structure of problem solving processes independently of its realization in animals" (McCarthy, 1974, p. 317). For yet another group of researchers, AI is really the study of human problem solving, concept learning, and language comprehension, and is a branch of theoretical psychology.

In fact, AI is all of these things. It presently encompasses many diverse areas of research which include the following major topics:

1. Problem Solving
   a. Game Playing
   b. Theorem Proving
   c. Heuristic Search
   d. Inference
2. Perception
   a. Pattern Recognition
   b. Scene Analysis/Description
   c. Context-based Vision
3. Natural Language Understanding
   a. Language Translation
   b. Question Answering
   c. Semantics
   d. Speech Understanding
4. Programming Languages
   a. Program Verification and Generation
   b. AI Languages
   c. Study of Programming Errors
5. Robotics
   a. Modelling and Planning
   b. Hand-Eye Co-ordination
6. Learning & Adaptive Systems
   a. Artificial Evolution & Brain Theory
   b. Concept Learning
   c. Socratic Systems

Within each of these areas, certain topics are more in current focus than others.[1] For example, within the area of Natural Language Understanding, relatively little attention is presently devoted to language translation, however,

---

1. Anyone who is familiar with AI research will also be aware that there is plenty of overlap and interdependence between the various subareas in the list, as well as some current research which doesn't really fit into any one of these categories.

semantics and speech understanding are in vogue. Furthermore certain areas are more in the spotlight than others, e.g., natural language is at present receiving considerable attention while robotics is not.

While the list above imparts some idea of the range of AI research, it does not convey the different perspectives which can characterize research within almost any one of these areas. Newell (1974) has discussed three different conceptual orientations to AI:

1) AI as the exploration of intellectual functions. The main question of this orientation is "What mechanisms can accomplish what intelligent functions?" (Whether this has anything to do with human mechanisms is irrelevant.) In this perspective would be included research on pattern recognition, theorem proving, problem solving, game-playing, etc.

2) AI as the science of weak methods. In this perspective, AI is a field devoted to the discovery and collection of a set of weak methods such as generate and test, heuristic search, hill climbing, hypothesize and match and others.

3) AI as theoretical psychology. AI serves as theoretical psychology if one adopts the view of human cognition as involving the processing of information represented in terms of discrete symbols and elementary operators/rules. Furthermore, in this perspective, the question of whether the program works in the same way humans do is of consequence.

It is possible to conduct research in any of the areas outlined above and within any of these 3 perspectives. Thus, depending upon one's orientation, AI could be properly considered a branch of computer science, psychology, mathematics, or even philosophy.[2]

In this paper the focus is on the educational applications of AI research, particularly the specific areas of research which have implications for computer assisted instruction (CAI). While AI and CAI are blood-brothers in the sense that they share a common computer methodology and a certain subset of common problems, there has tended to be little formal or informal interaction between the majority of professionals in both domains. Undoubtedly there are many reasons why this is so, however, I suspect one of the most major is that in essence AI is a fundamentally theoretical

2. In fact, recognition of the interdisciplinary nature of much AI research has led to the recent formation of a new field, Cognitive Science (and a journal of this name) which spans psychology, computer science, linguistics, philosophy, mathematics, and education.

endeavor and CAI is a practically oriented activity. Thus, things which are emphasized in doing AI work (e.g., generality, elegance, parsimony, etc.) are not valued as highly as certain pragmatic considerations in CAI (e.g., efficiency, costs, simplicity).[3]

As a first step in encouraging formal interaction between AI and CAI, this article discusses some current AI research and attempts to point out its relevance to CAI activity. The following sections will discuss research in natural language understanding, rule induction, programming languages and socratic systems. The last section outlines some of the ways in which AI research could be introduced into present CAI efforts.

I. Natural Language Understanding

It seems appropriate to begin with this area since it has a very evident and immediate application to CAI. Work on natural language (NL) understanding systems (also called computational linguistics) i.e., programs which can interpret stories, analogies, jokes, answer questions, and produce paraphrases, probably represent the major thrust of current AI research. Furthermore, I think it is fair to say that the work on NL systems constitutes the second major generation (or paradigm) of AI work; the first generation being mainly concerned with inventing very general methods (heuristics) for searching combinatorial spaces. This second generation of AI research focuses on problems concerned with the representation of knowledge.

Although there were important precursors, the first really significant work on NL systems was Winograd's (1972) "blocks world". Winograd's NL system was actually a single component of the robot SHRDLU which consisted of a simulated eye-hand manipulation system for moving toy blocks on a tabletop (e.g., cubes, cones, pyramids). The NL system was used to tell SHRDLU which blocks to move and to allow it to respond. SHRDLU was able to demonstrate its understanding of language by performing appropriate actions (or indicating that an action was impossible).

Winograd's NL system introduced some very important features for subsequent NL work. First, the system was programmed in Micro-Planner (a subset of PLANNER discussed later) and PROGRAMMER. Micro-Planner allowed the construction of the "blocks" database in which factual knowledge and causal reasoning about the blocks and their

---

3.However, as I have argued passionately elsewhere (Kearsley, 1977a), I think CAI ought to be following a more theoretical and less ad-hoc development.

environment was stated via procedural declarations. An example of such a procedural declaration for the definition of a cube is as follows:

```
(CUBE ((NOUN (OBJECT ((MANIPULABLE RECTANGULAR)
((IS ? BLOCK) (EQUIDIMENSIONABLE ?)))))))
```

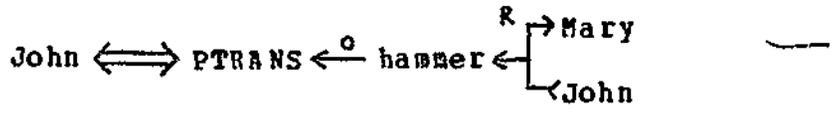
Actions were transformed into goal structures which made use of these procedural declarations. For example, in order to perform the action of grasping an object, the following goal stack might be created:

```
(GRASP B1)
    (GET-RID-OF B2)
        (PUT-ON B2 TABLE)
            (PUT B2 (435 201 0))
                (MOVEHAND (553 301 100))
```

which indicates that in order to achieve the goal of grasping object B1, first check to see if B1 is being grasped already, then check to see if anything else is being held (and if so GET-RID-OF it), which would require the goal of putting the currently held object (B2) on the table by moving the hand (the numbers indicate the coordinates).

PROGRAMMER was the language used to actually parse and generate language. It was based on a systematic grammar and involved a combined syntactic-semantic approach to parsing. The manipulations of SHRDLU and the table with the blocks were all simulated on a CRT screen. SHRDLU's vocabulary consisted of only a few hundred words specific to the "blocks world". The overall importance of Winograd's work was a clear-cut demonstration of the necessity of having a very detailed representation of relevant knowledge in order to understand language.

About the same time as winograd's work was published, Schank's (1972) conceptual dependency theory appeared. Schank's basic premise was that the key to language understanding involved an analysis of a small number of primitive actions such as grasp, move, transfer, etc. and that all NL parsing should be conceptually based around these primitives (hence conceptual dependency theory). An example of a simplified conceptual dependency diagram for the sentence, "John hit Mary with a hammer." is:



As this illustration shows, the analysis is built around the primitive act (in this case, physical transfer, PTRANS), the actors, and the object acted upon. This diagram can be mapped directly into a LISP statement and would represent the coded form of this sentence in terms of concepts.

More recent work by Schank and his students (Schank, 1975) has emphasized the notion of "scripts" which are

organized units of concepts about common or stereotyped occurrences and situations (e.g., ordering food in a restaurant, a bus trip, etc.). Scripts provide domain specific knowledge necessary for reasoning and inferencing which are essential in understanding language. Current work by Schank is focused upon large units of information such as stories, conversations, or paragraphs, in contrast to the early work which focused essentially on isolated sentences. Recent introductions to NL research are Charniak & Wilks (1976) and Chapter 6 in Raphael (1976).

The usefulness of this area of research to CAI is two-fold. First there is the possibility of replacing the standard keyword-based answer analysis used in almost all traditional CAI languages (e.g., COURSEWRITER, TUTOR, CAN) with a "real" analysis based on meaning. This would relieve the author of specifying complicated series of equivalent words, prohibited words, words to be ignored, etc., using AND/OR logic. All that would be required would be the specification of the appropriate concept and the system could determine correct paraphrases. This would not, however, relieve the author from specifying the boundaries on what counts as an acceptable answer, a partial answer, a misconception, etc. And since I think this constitutes the major work in constructing answer analysis, real understanding or comprehension would not really be much of an advantage. Thus, this aspect of the use of NL in CAI seems relatively minor.

The second use of natural language comprehension is more fundamental. As should be evident, in order to have the capability to understand and comprehend a student's response in a particular subject matter, a very complete and detailed representation of the subject matter must be programmed. This includes basic propositions, and their inter-relationships on the basis of categorical inclusion or via inferences/deductions. If this type of knowlege base is available, it is then possible to manipulate this knowledge via different learning strategies (i.e., simulations, games, drill & practice, socratic, etc.). Thus the actual knowledge would be essentially independent of the mode of instruction. Glimpses of exactly how this can be done are given in later sections of this paper. Suffice it to say here that the availability of such a knowledge base permits the CAI system to "understand" what it is teaching, to build a model of the student's state of understanding, and to identify deficiencies in its own understanding of a subject. Clearly this represents an impressive gain over our current generation of "non-intelligent" author languages and a step closer to providing the full sophistication of a good teacher.

Another aspect of AI research on natural language is work on speech understanding systems. Like the work on NL

systems, the work on speech understanding has used small and well-defined knowledge domains. For example, the HEARSAY system (Reddy & Newell, 1974) has been developed for the task domain of verbal chess moves; the SPEECHLIS system (Nash-Webber, 1975) involves the task domains of lunar geology and travel budget management. Research in speech understanding tackles the same sort of problems of semantics and knowledge representation which exist in NL research with the extra addition of a phoentic component.

While the practical application of speech understanding research is further removed than that of the NL systems, this research could be applied in the CAI domains of reading and language learning. More remote in time is the possibility of using verbal conversation and bypassing the necessity for written communication via keyboards. Not only would this revolutionize the nature of terminal construction and answer analysis procedures but it would also radically alter the nature of the instructional interaction, for now the student would be able to interrupt and interrogate the system during instruction.

While research in both natural language and speech understanding systems is still in its infancy, it could already be applied to CAI work. Many topics now taught via CAI would be ideal for exploring natural language interaction or speech recognition. Furthermore, CAI provides an excellent testing ground for AI systems with its real-world problems and difficulties.

Rule Induction

The study of rule formation and utilization in solving problems, concept learning, and understanding language has emerged as an important area of AI research and overlaps with research in cognitive psychology. In so far as CAI depends upon or embodies a theory of instruction which is based on the cognitive processes involved in the acquisition of knowledge, research on rules is of considerable relevance to CAI. In addition, rules have been emphasized in the context of instruction in the work of Gagne (e.g., Gagne, 1966).[4]

The work of Egan & Greeno (1974) and Simon & Lea (1974) are attempts to make comprehensive theoretical statements about rule induction across the tasks of concept learning, serial pattern learning, and problem solving. The basic premise of these statements is that the knowledge structures

---

4. Elsewhere (Kearsley, 1976), I have discussed the possibility of using rule structures to model individual differences in an educational context.

used and acquired in performing these tasks are rules stored in working memory. Such rules are abstracted or induced by observing one or more examples of their use. In the Egan & Greeno analysis, attention is paid to three distinctive issues in rule induction: the kind of knowledge structure which is constructed as a result of experience in a task (and hence enables performance), the kinds of knowledge structures that individuals need as prerequisites for learning rules, and how the process of acquisition (of rules) occurs. For concept learning they conclude that the acquired structure is a decision tree with nodes representing tests of attributes for presented stimuli; for serial patterns, the general form of the acquired structure is a hierarchy of operators defined on an alphabet of some sort; and in the case of problem solving, that the general form of the acquired structure is a hierarchy of goals (or goal structure) involving transformations of the specific features of the problem situation. Table 1 is their concise summary of their analysis of the three aspects of rule induction for concept learning, serial patterns, and problem solving.

Insert Table 1 here.

Simon & Lea propose that rule induction tasks (i.e., concept learning, serial patterns) and problem solving can both be interpreted in terms of problem spaces and information processes for searching such spaces. Their major conclusion is the distinction between rule induction tasks and problem solving tasks is that the former require two problem spaces (one for rules and one for instances) while problem solving requires only one space. Thus in a rule induction task, the attainment of a solution is determined by simultaneously finding an appropriate rule and testing all instances with it; in the case of problem solving the solution is achieved when a set of rules which produce the desired result is found.

Another line of AI research which contributes to the study of rule induction is Winston's (1975) research on the learning of concepts in the "blocks world". The system was able to induce simple concrete concepts of structures (e.g., arch, pedestal, house,etc.) from "seeing" instances and non-instances composed on block structures it knew about. The basis for Winston's system was a network notation (implemented in LISP) which allowed the representation of concepts and their comparison. The crucial procedure in the learning of concepts was the analysis of "near misses" of a concept in order to abstract the essential (and identify non-essential) attributes. While the details of how the system worked are fascinating, it is sufficient for our purposes to note the importance of negative instances in concept learning.

TABLE 1

Summary of Analysis of Egan & Greeno (1974).

Kind of rule acquired	Acquired structure	Prerequisite knowledge	Process of acquisition
Conceptual Classification:	Decision tree		
Associative grouping (verbal concept)	Find which category feature is present.	Associations in semantic memory	Scanning instances and noticing common associations
Single-feature classification	Test single feature	Knowledge of or ability to detect features	Selection and elimination of features
Multiple attributives (connective structure known)			
Connective structure (attributes known)	Conditional feature tests	Truth table combinations	Selection of rules or association of response to combinations of features
Complete classificatory rule		Features and combinations	Construction and modification of decision tree
Relational concepts	Test n-ary relational feature	Knowledge of or ability to detect relations (relations in semantic memory for verbal materials)	Search in set of known relations, within subsets ordered by complexity
Analogies	Conjunction of relational features		Note relations and test on alternative answers
Sequential concepts:	Tree structures of relations	Knowledge of alphabet, and detection or computation of relations	
Bt chunks	Sequence of chunks	Same, next, and backward-next on adjacent elements	Scan sets of elements to identify chunks
Biork sequences	Sequence of interval relations	Difference between adjacent elements, and same or next relations on differences	Compute and store sequence of differences; compare entries to detect advancing rules
Thurstone sequences	Interleaved sequences of interval relations	Same, next, and backward-next on adjacent and separated elements	Find period of sequence, formulate rule, and test on elements
Restle hierarchical sequences	Structure of relations between subsequences	Differences between adjacent elements and between subsequences	Identify subsequence chunks and transformations between subsequences
Solution patterns for solving problems	Structure of transformations defined on sets of states	Space of problem states and set of transformations	Induce pattern in sequence of transformations, including modification of differ[illegible] ordering and connect[illegible] formations to feat[illegible]; modify encod[illegible] problem, if necessary

Yet another important line of research on rule induction is the study of causal reasoning and inference mechanisms in connection with natural language work. A good example is the work of Schank & Reiger (1974). Schank & Reiger identified 12 types of inferences which could be used during or after the conceptual analysis of a sentence. For example, an instrumental inference would occur when a primitive act has been referred to and a probable instrument of that act is inferred. Abelson (1974) in his analysis of belief systems has extended the study of inference mechanisms beyond the linguistic level to include inferences about other peoples' intentions, roles, and plans. Reiger (1976) has attempted to show how "commonsense alogoritmic knowledge" underlies both language comprehension and problem solving.

The implications of this work on rule induction are as follows. Problem solving and concept learning are 2 basic components of learning and hence an understanding of the underlying cognitive processes will be essential contributions to a knowledge of how to teach these components. In the case of concept learning, this research suggests that the presentation of concepts in the form of decision trees with emphasis on positive and negative instances will optimize learning. The importance of identifying positive and negative attributes has often been stressed by educators (e.g., Becker, Englemann, & Thomas, 1975; Markle & Tiedmann, 1972). For problem solving, the importance of subgoal generation and means-ends anlysis in terms of finding a set of rules which transform the problem space suggests the importance of hierarchical analysis and presentation of problems.

The explicit analysis of rule induction in the domains of concept learning and problem solving, particularly their expression in computable formalisms (e.g., the General Rule Inducer of Egan & Greeno), provides the possibility of building these rules into a CAI tutor. Thus in the case of problem solving in a programming lab for example, it would be possible to have the system test or guide the problem-solving of a student by actually solving the problems itself (and hence generating the necessary subgoals) or alternatively, generate problems according to certain specifications. This is of course quite close to CAI research in generative programming (e.g., Koffman & Blount, 1976). In the case of teaching concepts, such a system could arrange suitable instances to build a satisfactory decision tree representing a concept.

Finally, a computational understanding of inductive inferences, such as that emerging from current natural language research has applications in the automatic generation of hypothesises and questions. This is an essential component of socratic (dialog) strategies and also

would alleviate the need for the programming of fixed evaluation seguences for all students. Furthermore, this capacity would be extremely valuable for the generation of questions for domain or criterion referenced evaluation (see Anderson, 1972 or Bormuth, 1970).

Programming Languages

AI research has probably catalyzed more novel software development than any other contemporary branch of computer science. In particular, AI work provided the impetus for the development of families of non-numerical languages such as LISP, LOGO, and more recently, entire subspecies of goal-driven problem solving languages such as PLANNER, CONNIVER, QA4 (now QLISP), SAIL, and POP2 (see the summary by Bobrow & Raphael, 1974). In addition to providing the stimulus for the development of new programming languages, AI research includes the exploration of automatic programming and program proving (verification) and also the study of programming errors and debugging. All of these different lines of research in programming have relevance to CAI efforts.

AI programming languages (particularly LISP and its derivitives) provide very different alternatives for the basic organization of CAI courseware. All major CAI authoring languages presently in use (e.g., TUTOR, COURSEWRITER, CAN) are "frame-oriented" (FO) in that they provide for a basic instructional pattern:

1) Presentation of text
2) Posing a question/problem
3) Answer analysis
4) Contingent branching

These four aspects comprise a single instructional "frame" (even though they may actually span many physical frames). Regardless of the initial orientation of the programmer, all instruction will eventually adhere to this basic pattern since in fact this constitutes the basic data structure of a FO language. Notice also that in a FO language instructional content and instructional logic are non-separable and together comprise the data structure.[5]

In contrast, the use of LISP-like languages necessitates the use of a semantic network (SN) type of data structure since the basic list structure requires tree-like or network definitions.[6] The advantage of a SN type of data structure is that a sophisticated, meaning-based language

---

5. For a similar argument from an information retrieval perspective, see Osin (1976).
6. Introductions to LISP are given by Siklossy (1976) and Winston (1977).

analysis is permitted; that the database can be interrogated as well as presented; and, most importantly, the system is "intelligent" in the sense that it can trace its own actions. Furthermore, SN languages allow a separation of content and logic. These points will be elaborated upon in the following discussion of student models and later in the section on socratic systems.

Self (1974) provides a demonstration of how AI languages and methods can be used to implement "student models", i.e., representations of the hypothesized knowledge states of the students. Use of a student model allows the course of instruction to be guided by the specific comparison between the student's state of understanding of a subject and a complete understanding of the subject matter. Thus, individualization of instructional content is not handled via branches contingent upon quantitative evaluation (i.e., number of correct answers) but rather pattern-directed evocation of certain knowledge based upon qualitative inferences about what the student understands or misunderstands. This is essentially equivalent to building in the criteria and inferences that a human instructor uses when setting up branching conditions in a traditional FO language.

Self uses a version of the goal-directed PLANNER language to illustrate a student model. In PLANNER, information is stored in basically two forms: facts written as symbolic expressions, and theorems which are symbolic procedures for deducing facts from other facts. Thus, a PLANNER database consists of two types of knowledge: facts and procedures. Multiple databases may be created allowing the construction of plans and hypothetical situations. Execution procedes via the computation of goal structures. In order to achieve such goal statements, PLANNER automatically searches the database(s) to try and find matching goal patterns (the so-called "pattern-directed invocation" feature).

Insert Figure 1 here.

A fragment of PLANNER code from Self's program is shown in the bottom of Figure 1; this corresponds to the dialogue in the top part of the Figure. While this may be incomprehensible to anyone unfamiliar to LISP-like languages, some of its major components can be pointed out. Statements 2 and 3 define the facts that Montana is further inland than Newfoundland and that Newfoundland is further north than Montana. Statement 4 is a theorem (inferential rule) which indicates that X is colder than Y if X is further north or further inland. Statements 6-9 represent another theorem which indicates how to derive a contradiction between facts X,Y, and Z in databases A and B.

1. Teacher: Which is likely to be colder, Newfoundland or Montana?
2. Student: Newfoundland.
3. Please give your reasons for answering Newfoundland.
4. Newfoundland is further north.
5. Yes, that's true. Are you arguing then, that if you take any two places in the Northern Hemisphere, the one which is further north will have the lower average temperature?
6. Yes, I guess so.
7. I'll tell you now that Montana has lower average temperatures than Newfoundland. Does this fact cause you to change your reasoning?
8. Yes.
9. In what way?
10. Being further north isn't as important as I thought.
11. Please try to be more precise. Would you, for instance, say that if you take any two places in the Northern Hemisphere, the one furthest south has the colder temperature?
12. No, I wouldn't say that.
13. Would you say that how far north a place is has no effect on temperature?
14. No, I know it has some effect.
15. Yes, you're right. Distance north does affect temperature. Yet, even though it is further north, Newfoundland has higher average temperatures than Montana. What must this mean?
16. There must be some other factor.
17. Yes! Right! What could this factor be?
18. I don't have any idea.

```
1. (PROG (MT MS X Y Z))
2. (LET F1 FACT (FURTHERINLAND MONTANA NEWFOUNDLAND))
3. (LET F2 FACT (FURTHERNORTH NEWFOUNDLAND MONTANA))
4. (LET T1 INFER (X Y) (COLDER ?X ?Y)
5.     (FACT (FURTHERNORTH ?X ?Y))
       (FACT (FURTHERINLAND ?X ?Y)))
6. (LET T2 INFER (X Y Z M1 M2 A B)
7.   (CONTRADICTION ?X ?Y ?Z ?M1 ?M2 ?A ?B)
8.     (DB ?M2) (GOAL (?X ?Y ?Z) (USE ?B))
9.     (DB ?M1) (NOT (GOAL (?X ?Y ?Z) (USE ?A))))
10. (LET S1 INFER (X Y) (COLDER ?X ?Y)
11.   (FACT (FURTHERNORTH ?X ?Y)))
12. (DB ?MT)
13. (ASSERT F1 F2 T1 T2)
14. (DB ?MS)
15. (ASSERT F1 F2 S1)
16. (DB ?MT)
17. (GOAL (CONTRADICTION ?X ?Y ?Z ?MT ?MS T1 S1) (USE T2))
18. (PRINT IS ?Y ?X THAN ?Z)
```

Figure 1. Fragments of dialogue and PLANNER code from Self (1974).

Statements 10 and 11 are an assertion (fact) that something must be cold if it is further north. Statements 12 and 13 places the first two facts and theorems in the database of the teacher; statements 14 and 15 put the two facts and first theorem in the student's database. Thus the student is assumed to be missing the second inferential rule about temperature and inland position. Statements 16 and 17 specify that a goal search is be initiated (directed to the student) for the facts which satisfy the contradiction given by the second theorem. If found, this is to be asked as a question. This corresponds to the pedagogical strategy shown in the dialogue in the top part of the Figure, i.e., attempting to get the student to learn both determinants of temperature. When the student can satisfy this sequence, it can be inferred that the student's knowledge encompasses this theorem or rule.

While the details of this example may be rather vague, it should be evident that this type of approach to CAI would require the very detailed specification of facts, concepts, principles, their interrelationships, and inferences which can be made between them. Doing so allows the development of an "intelligent" system in which the meaning of the student's reponses are used. This example also illustrates how content (i.e., facts and theorems) and instructional logic (goal structures) can be keep separate in a PLANNER-like language.

Considerable research has been devoted to the study of programming errors and debugging processes (e.g., Gould, 1975; Mayer, 1975), probably inspired by Weinberg's pioneering efforts (Weinberg, 1971). This research has encompassed the study of the frequency of different types of programming errors, the difference between the errors made by experienced and inexperienced programmers, the effects of providing models or "mental sets" during programming activity, and the type of strategies and tactics used in debugging programs. For a recent review of this research, see Kearsley (1977b).

This research has obvious relevance to the teaching of programming and the construction of problem-solving laboratories in programming. The work of Barr et al. (1976) is particularly interesting in this connection because they have developed a curriculum which combines elements of AI techniques with aspects of instructional design. Their BASIC Instruction Program (BIP) involves a special BASIC interpreter written in SAIL which provides for a curriculum information network composed of elementary programming skills. BIP does not follow a fixed presentational sequence but selects problems via a task selection algorithm which chooses problems incorporating skills that a particular student appears to be lacking. Figure 2 is a flowchart which illustrates the logic of the task selection algorithm.

Insert Figure 2 here.

The BIP interpreter also involves an error diagnosis system which is tutorial in nature, although this does not appear to be as sophisticated as that of Wilson et al. (1976) in the PLATO CAPS system.

Work on automatic program generation (also called program synthesis) and program verification (e.g., Simon & Siklossy, 1972) involves the attempt to design programs which can actually generate other programs according to specifications and programs which are capable of proving that a program does what it is intended to do (as opposed to determining this inductively via debugging). The facility to take a student's program and verify its correctness or to generate programs according to a student's specification for an example would be very useful in programming problem solving labs. In the generative programming of Koffman & Blount (1976), certain capabilities for program synthesis and verification appear to have been provided.

LOGO represents another AI alternative to FO author languages. LOGO is an extremely simple and powerful language which is fully extensible and recursive. Along with LOGO goes a particular "lassiz - faire" approach to providing a learning environment as exemplified by the work of Papert (e.g., Papert & Solomon, 1972). The important technique provided by LOGO is that of a procedure -- namely the understanding of something in terms of a computable expression. Papert (1971) has argued this approach most strongly in the domain of teaching mathematics to children. An interesting application of LOGO to autistic children has been reported by Emanuel & Weir (1976).

To summarize this section, AI programming languages provide some alternative approaches to the usual instructional logic of CAI author languages. This includes the semantic network orientation of LISP-like languages, the problem-solving and inferential capabilities of the PLANNER-like languages, and the procedural orientation of LOGO. In addition, research on programming errors and program synthesis / verification has many potential applications in the teaching of programming languages.

Socratic Systems

We come now to the area of AI research which is closest to CAI activity in the sense that it is explicitly concerned with instruction. This is the work on socratic systems,

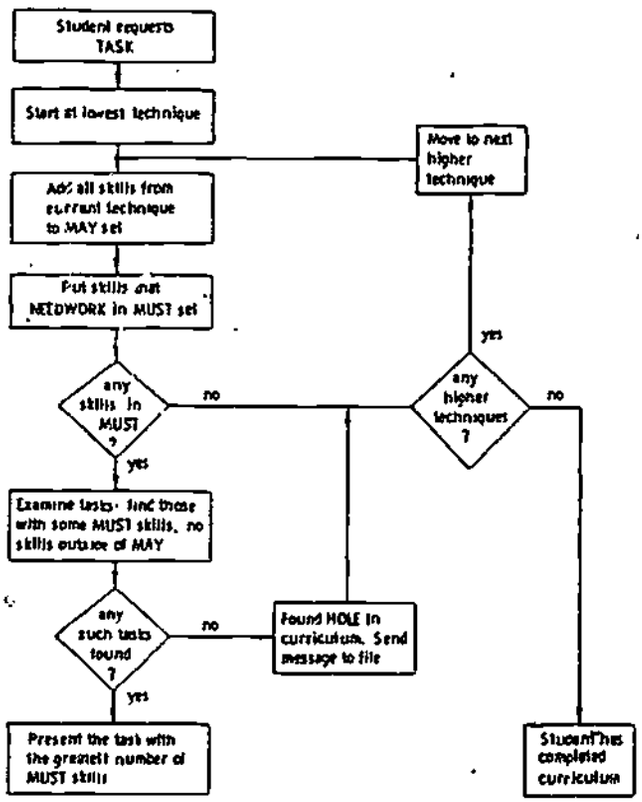


Figure 2. Task Selection Algorithm of Barr et al. (1976).

mostly conducted by members of Bolt, Beranek & Newman.[7] The basic idea of a socratic system is that information exchange should proceed via dialogue and questioning intitiated by either the teacher or the student (hence it is also called "mixed-inititive" instruction). Socratic dialogue is the paradigm case of individualized instruction and also is the most demanding of instructional methods for both teacher (knowledge and skills) and student (attention, motivation). Hence, the implementation of a socratic system is a very challenging problem in artificial intelligence.

The first attempt to implement a socratic system was made by Carbonell (1970) in the program SCHOLAR and the task domain of geography. Carbonell pointed out the difference between the semantic-network approach and the "generative" CAI approaches. Generative CAI depended upon algorithmic generation and manipulation and therefore excluded the possibility of verbally-oriented subject matters whereas a semantic-network approach suffered no such limitation. In fact, Carbonell's rationale for choosing geography was that it represented a verbally-oriented subject with essentially no algorithmic structure. Carbonell also pointed out the potential contribution of a socratic system to the understanding of errors and misconceptions, and the eventual need to develop general instructional strategies to handle missing and partial information, overgeneralizations, confused facts, etc. Because of the way answer analysis is handled in traditional CAI languages, no such interest in a general theory of errors has emerged from CAI.

Work on SCHOLAR has been continued by Collins and colleagues (e.g., Collins, 1976; Collins, Warnock & Aiello, 1975; Collins, Warnock & Passafiume, 1974). Much of this work has involved the study of the cognitive processes involved in socratic dialogues (i.e., reasoning from incomplete knowledge, types of inferences, questioning strategies, reviewing, providing hints, error correction, hypothesis formation, etc.). The most recent effort (Collins, 1976) is an attempt to formulate rules which produce socratic strategies. The rules are formalized as a production system which could be programmed.

While Collins has been devoting attention to the theoretical aspects of socratic dialogues, a production version of SCHOLAR has also been implemented. The NLS-SCHOLAR system (Grignelli, Gould & Hausman, 1975) is a socratic system for teaching individuals how to use the NLS

7. Socratic systems are also called tutorial systems but I shall stick to the label "socratic" since the term "tutorial" has a somewhat different meaning in a CAI context.

text editor. Most interesting about the report of their project is their discussion of the problems encountered in the initial implementation of the system. This included problems such as the handling of spelling errors, unanticipated synonyms, irregular syntax, lack of program knowledge, poor answers to questions, and unanticipated contexts. There were also numerous problems related specifically to the use of natural language such as anaphoric references, ellipses, indeterminate references, and paraphrase equivalence -- all familiar problems in natural language research. Here we see the need for pooling of conceptual resources between those working on natural language and socratic systems.

Another implementation of a socratic system is SOPHIE (Brown & Burton, 1975) which involves an electronics database. The nature of SOPHIE is similar to the problem solving system of Barr et al. and Wilcox et al., except that the task involves the debugging of malfunctioning electronic equipment rather than computer programs. SOPHIE implements many ideas on hypothesis and question generation and also involves a dynamic simulator which generates and checks information for questions and answers.

The work on socratic systems incorporates many of the various areas of AI research previously discussed: natural language understanding, rule induction and inferences, studies in programming errors and programming languages. In my opinion, it comes the closest to providing truly individualized instruction since both the mode and content of instruction are tailored to the individual.

Conclusions

This article has surveyed some major areas of AI research and discussed their potential importance to CAI activity. There are other areas of AI research which may have eventual relevance to CAI. For example, it has been suggested that answer analysis can be considered a problem similar to those faced in pattern recognition, namely, determining whether a particular answer matches a desired answer. However, in terms of any immediate interaction between AI and CAI, the four areas of natural language, rule induction, programming, and socratic systems are most likely to lead to an interchange of ideas. Possible applications of the research in these four areas has been specifically mentioned in each section. By way of conclusion, we will consider the general problem of applying AI work to CAI.

Individuals involved in CAI represent heterogeneous backgrounds and perspectives and hence have differing orientations to CAI activity. Thus, just as AI means different things to different people so CAI represents a variety of purposes. One group are the media and

instructional specialists who see CAI in terms of its potential for research in instructional design. Another group includes psychologists who consider CAI as a fertile medium or tool for the study of cognitive processes, individual differences, testing procedures, etc. Those involved in CAI from a computer science background tend to see CAI in light of information retrieval, data structures, or other software/hardware characteristics. Finally, there are educators from many different fields (e.g., medicine, chemistry, physics, etc.) who have no interest in CAI itself, except in terms of their own subject areas. To this latter group (which is probably the largest proportion of CAI authors), CAI is really no more than a sophisticated delivery system.

This wide range in orientations to CAI means several things. First, different lines of research in CAI will appeal primarily to certain orientations, e.g., programming studies to those with a computer background. Secondly, many features of CAI languages and systems exist because they satisfy the requirements of certain groups. Thus, in terms of delivering instruction, it is important that the system be reliable and provide quick response time. Despite the theoretical inelegance of traditional frame-oriented CAI author languages, they are easily programmed by authors with no prior programming experience. Furthermore, if CAI systems are to be instructionally effective, they must provide adequate graphic capabilities and automatic collection and processing of student performance data. It should be evident that most AI research which is relevant to CAI work (e.g., socratic systems) is not concerned with the type of features which are very important from an educator's point of view.

So something of a dilemma exists in terms of the exploration of AI techniques and ideas in a CAI context. Unfortunately, the existing tendency in CAI seems to be to accept this problem as an excuse not to pursue research into alternate CAI methods. This is most unsatisfactory when the very rudimentary and primitive state of current CAI systems is considered. Furthermore, it seems distinctly embarassing to me that some of the most important research in CAI is presently being done in the field of AI not CAI.

What could or should be done? Some possibilities include:

*joint conferences which bring together workers from AI and CAI. CAI workers can become aware of the research in AI and AI people can be exposed to the variety of educational applications

*creation of CAI projects to explore certain AI methods, languages, etc. in a research (rather than production) environment

*selection of instructional contexts and applications for research in AI

*joint AI-CAI research projects, encouraged by shared funding

Some may feel that interactions between AI and CAI will occur naturally without any explicit need for planned interaction. While I would like to believe this, I suspect that without some deliberate match-making, AI and CAI will proceed in blissful ignorance of each other.

References

Abelson, R.P. The structure of belief systems. In Schank & Colby (Eds.).

Anderson, R.C. How to construct achievement tests to assess comprehension. Review of Educational Research , 1972, 42 , 145-170.

Barr, A., Beard, M., & Atkinson, R.C. The computer as a tutorial laboratory: the Stanford BIP Project. Man-Machine Studies , 1976, 8 , 567-596.

Becker, W.C., Englemann, S., & Thomas, D.R. Teaching II: Concept Learning. Chicago: Science Research Associates, 1975.

Bobrow, D.G. & Collins, A. Representation and understanding: Studies in cognitive science. New York: Academic Press, 1975.

Bobrow, D.G. & Raphael, B. New programming languages for AI research. ACM Computing Surveys , 1974, 6(3) , 153-174.

Bormuth, J.R. On the theory of achievement test items. Chicago: University of Chicago Press, 1970.

Brown, J.S. & Burton, R.R. Multiple representations of knowledge for tutorial reasoning. In Bobrow & Collins (Eds.).

Carbonell, J.R. AI in CAI: An artificial intelligence approach to computer assisted instruction. IEEE Transactions on Man-Machine Systems , 1970, 11 , 190-202.

Charniak, E. & Wilks, Y. Computational Semantics. New York: Elsevier, 1976.

Collins, A. Processes in acquiring knowledge. Bolt, Bernak & Newman TR-3231, January 1976.

Collins, A., Warnock, E.H., Aiello, N., & Miller, M.L. Reasoning from incomplete knowledge. In Bobrow & Collins (Eds.).

Collins, A., Warnock, E.H., Passafiume, J.J. Analysis and synthesis of tutorial dialogues. Bolt, Bernak & Newman TR-2789, March 1974.

Dwyer, T.A. Heuristic strategies for using computers to enrich education. Man-Machine Studies , 1974, 6 , 137-154.

Egan, D.E. & Greeno, J.G. Theory of rule induction: knowledge acquired in concept learning, serial pattern learning, and problem solving. In L. Gregg (Ed.).

Emanuel, R. & Weir, S. Catalysing communication in an autistic child in a LOGO-like learning environment. Proceedings AISB Summer Conference , Edinburgh, July 1976.

Gagne, R.M. The learning of principles. In H.J. Klausmeier & C.W. Harris (Eds.), Analyses of concept learning. New York: Academic Press, 1966.

Gould, J.D. Some psychological evidence on how people debug computer programs. Man-Machine Studies , 1975, 7 , 151-182.

Gregg, L. Knowledge and cognition. New York: Wiley & Sons, 1974.

Grignetti, M.C., Gould, L. & Hansmann, C. NLS-SCHOLAR: Modifications and field testing. Bolt, Bernak & Newman TR-75-358, 1975.

Kearsley, G.P. Individuality, individual differences and computer simulation. Educational & Psychological Measurement , 1976, 36 , 811-823.

Kearsley, G.P. Some conceptual issues in CAI. Journal of Computer Based Instruction , in press, 1977a.

Kearsley, G.P. Computer programming as a cognitive process. Manuscript submitted for publication, 1977b.

Koffman, E.B. & Blount, S.E. Artificial intelligence and automatic programming in CAI. Artifical Intelligence , 1975, 6 , 215-234.

Luehrmann, A.W. Should the computer teach the student or vice-versa? Creative Computing , November 1976, 42-45.

Markle, S.M. & Tiemann, P.W. Conceptual learning and instructional design. Journal of Educational Technology , 1970, 1 , 52-62.

Mayer, R.E. Different problem-solving competencies established in learning computer programming with and without meaningful models. Journal of Educational Psychology , 1975, 67 , 725-734.

McCarthy, J. Book review of the Lighthill Report. Artificial Intelligence , 1974, 5 , 317.

Nash-Weber, B. The role of semantics in automatic speech understanding. In Bobrow & Collins (Eds.).

Newell, A. Artificial Intelligence and the concept of mind. In Schank & Colby (Eds.).

Osin, L. SMITH: How to produce CAI courses without programming. Man-machine studies , 1976 8 , 207-241.

Papert, S. Teaching children to be mathematicians vs. teaching about mathematics. MIT AI Memo 249, July 1971.

Papert, S. & Solomon, C. Twenty things to do with a computer. Educational Technology , 1972, 12 , 9-18.

Peele, M.A. Computer glass boxes: Teaching children concepts with "A Programming Language (APL)". Educational Technology , 1974, 14 , 9-16.

Raphael, B. The thinking computer: Mind inside matter. San Francisco: Freeman, 1976.

Reddy, R. & Newell, A. Knowledge and its representation in a speech understanding system. In L. Gregg (Ed.).

Reiger, C. An organization of knowledge for problem solving and language comprehension. Artificial Intelligence , 1976, 7 , 89-127

Schank, R. Conceptual dependency: A theory of natural language understanding. Cognitive Psychology , 1972, 3 , 552-631.

Schank, R. Conceptual Information Processing. Amsterdam: North Holland, 1975.

Schank, R. & Colby, K. Computer models of thought and language. San Francisco: Freeman, 1974.

Schank, R. & Reiger, C.J. Inference and the computer understanding of natural language. Artificial Intelligence , 1974, 5 , 373-412.

Self, J.A. Student models in computer assisted instruction. Man-machine Studies , 1974, 6 , 261-276.

Siklossy, L. Lets talk LISP. Englewood Cliffs, N.J.: Prentice-Hall, 1976.

Simon, H.A. & Lea, G. Problem solving and rule induction: A unified view. In Gregg (Ed.).

Simon, H.A. & Siklossy, L. Representation and meaning. Englewood Cliffs: Prentice-Hall, 1972.

Uttal, W.R. et al. Generative computer assisted instruction. Newburyport, Mass.: Entelek, 1970.

Weinberg, G.M. The psychology of computer programming. New York: Van Nostrand Reinhold, 1971.

Wilcox, T.R., Davis, A.M., & Tindall, M.H. The design of a table driven, interactive diagnostic programming system. Communications of the ACM , 1976, 19 , 609-616.

Winograd, T. Understanding natural language. New York: Academic, 1972.

Winston, P. Learning structural descriptions from examples. In P. Winston (Ed.), The psychology of computer vision. New York: Academic, 1975.

Winston, P. Artifical Intelligence. Mass.: Addison-Wesley, 1977.