Artificial Intelligence and Computer-Assisted Language Instruction: A Perspective

Alan Bailin
University of Western Ontario

ABSTRACT: The article attempts to outline the major components of CALI-AI (computer-assisted language instruction incorporating artificial intelligence techniques). The article begins by discussing briefly the central assumption on which CALI-AI work is based, that is, that human cognitive abilities can be reproduced by mechanical means. It then proceeds to examine the following components of CALI-AI: (1) natural language processing, (2) problem solving, (3) language learning, and (4) modeling teacher behavior. The article concludes with a discussion of the ways in which language teachers can participate in the development of the field.

KEYWORDS: artificial intelligence, computer-assisted language instruction, natural language processing, language teaching.

Introduction

This article attempts to outline the major components of CALI-AI (computer-assisted language instruction incorporating artificial intelligence techniques) from the perspective of how these components are used at present and how they might be used in the future. In so doing, it tries to show that CALI-AI has features that distinguish it from other AI applications and that can allow it to make its own distinct contributions both to CALI and to AI.

The article begins by discussing briefly the central assumption on which CALI-AI work is based, that is, that human cognitive abilities can be reproduced by mechanical means. It then proceeds to examine the following components of CALI-AI: (1) natural language processing, (2) problem solving, (3) language learning, and (4) modeling teacher behavior. The article ends with a discussion of the ways in which language teachers can participate in the development of the field.

Human Cognition and Computability

The ultimate goal of CALI-AI is to model in a robust way the cognitive behavior of humans in a particular social role: that of language teacher. At least
in this regard, CALI-AI is not simply an attempt at sophisticated programming. It is, above all, an attempt to achieve a true AI goal: the replication by machine of significant aspects of human cognitive abilities. To test whether or not a machine could replicate human cognitive behavior, Alan Turing suggested that a human should interact with it without any knowledge about whether or not s/he was talking to a machine. If the human believed s/he was talking to another human, the machine could be considered "truly" intelligent. This test is known as the Turing Test.

The test would have no validity in relation to many AI systems designed for purely military or industrial purposes because these systems do not really aim at simulating human behavior. Rather, they are intended simply to aid in the making of purely technical decisions (the correct mixture of ingredients in a Soup mix, or the correct technical response to incoming missiles—see Buchanan 1985 for examples). On the other hand, if CALI-AI ultimately does achieve its goal, it should be able to pass the Turing test, because it will have successfully replicated a significant aspect of human behavior—that of a language teacher. A truly successful system would behave in ways indistinguishable from that of a human performing the same teaching function.

We are far from achieving this goal. Current CALI-AI projects cannot and, in the author's opinion, should not be used in place of a teacher. They are truly a part of CAI—computer-assisted instruction. Nevertheless, even at this stage, a great deal of what a teacher does can be replicated by machine. CALI-AI can check the syntax of a student's written work, create environments in which students use language in pedagogically beneficial ways, and provide sophisticated feedback to students engaged in drill-and-practice exercises.

Underlying both CALT-AI's small but significant actual contributions and its potential contributions is the question of what aspects of human cognitive behavior a computer can replicate. In the most general terms, the answer is that a machine can replicate any aspect of human behavior which can be represented or simulated by computational means. It must be stressed that "computational" here does not really mean involving numbers (not at least in an arithmetic sense). Rather, it refers to anything which can be described in terms of a "Turing machine."

A Turing machine is not a "real" machine, but rather an automaton, that is, an idealized abstract model of a machine. It is specifically intended not for numerical operations (although it can be used to compute them), but rather for the general manipulation of symbols. A Turing machine (see figure 1) consists of
(1) a finite number of states, (2) a tape of infinite length, (3) a finite number of tape symbols (including the blank (B) symbol), and (4) a tape head. Each tape cell contains only one tape symbol, and the tape head scans only one cell at a time. The machine can perform the following kinds of operations: (1) it can erase a tape symbol on the cell which the tape head is scanning and replace it with a non-blank tape symbol, (2) it can move the tape head one cell to either the left or right, (3) it can change 4) it can "halt" (that is, stop) completely (Hopcraft and Ullman 1969, 80ff and Partee 1978, 162ff).

![The Turing Machine Diagram](image)

**Figure 1: The Turing Machine**

The basic assumption of AI in general, and CALI-AI in particular, is that human cognition—or at least a significant portion of it—can be replicated by means of the plodding step-by-step moves of a Turing machine. Underlying CALI-AI then is not some magician's hocus-pocus, but rather the careful, precise analysis of what language teaching involves. Whether or not the basic assumption proves tenable, the attempt to develop CALI-AI should lead us to a better understanding of what it means to teach. As we examine the components of CALI-AI, this should be kept in mind.

**Natural Language Processing**

The field of natural language processing can be divided into the following areas: syntax, semantic/pragmatics, morphology, speech processing, and language generation. Below, each will be examined in turn.

**Syntax**

The two basic areas in which syntax is important in natural language processing are parsing and language generation. This section concerns only
parsing because, unlike language generation, it is an area of application where
the syntax operates to a large degree independently of other natural language
processing components (semantics, morphology, etc.).

The considerable work that has been done in natural language processing
has led to a variety of approaches and, as a consequence, a number of different
ways of categorizing parsers (see table 1). These categorizations provide us with a
way of exploring the properties of parsers.

<table>
<thead>
<tr>
<th>How They Parse:</th>
<th>Top-Down, Bottom-Up, Wait-and-See Parser (WASP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>How They Explore:</td>
<td>Backtracking, Parallel Parsing</td>
</tr>
<tr>
<td>Formal Grammar:</td>
<td>Type 0, context Sensitive, Context Free, Regular Grammar</td>
</tr>
<tr>
<td>Linguistic Grammar:</td>
<td>Government-Binding (GB), Lexical-Functional Grammar (LFG), Structure Grammar (GPSG)</td>
</tr>
</tbody>
</table>

Table 1

One way of classifying parsers is in terms of how the parsing procedure
operates. A top-down parser begins with the major syntactic units of a sentence,
then tries to find the immediate constituents of each of these, and so on until the
word units are reached. A bottom-up parser, on the other hand, tries to build the
structures from the word level up to the sentence level (see Grishman 1986, 27
and Winograd 1983, 90-91). A wait-and-see-parser (WASP) does not try to build
a major category from the start, but, as the name implies, waits until it has the
constituents necessary for making an identification. In other words, for the most
part, it tries to take the guesswork out of parsing (see Winston 1984, 309ff and
Marcus 1980).

However, even WASPs must guess occasionally, and like other parsers,
need to explore more than one syntactic analysis before deciding on an
appropriate parse. Parsers can be classified in terms of how they explore these
possibilities. A parser is said to backtrack if it explores one possible syntactic
structure after another until it finds the one which is required. Parallel parsing,
on the other hand, means that the parser explores all the alternatives at the same
time (see Grishman 1986, 27f and Winograd 1983, 368-369).

Parsers can also be classified in terms of the formal grammar type with
which they can be identified (i.e., can be considered mathematically equivalent
to). Formal grammars are described using "productions." These are rules which
take the following form:

\[ A \rightarrow B \]

A rule of this form is understood to mean that \( A \) consists of \( B \).
Classification in terms of formal grammar types relates to restrictions on what symbols can go on the left and right sides of the arrow. Context-sensitive grammars are grammars in which the right side of the rule (the $B$) must contain at least as many symbols as the left side ($AB \rightarrow BC$ and $AB \rightarrow BCD$, but not $AB \rightarrow B$). In context-free grammars, the left side must contain only one symbol and the right can have any number as long as it is not solely comprised of the symbol for "the empty sentence" ($A \rightarrow B$ or $A \rightarrow BCD$, but not $A \rightarrow \{\text{empty}\}$ or $AB \rightarrow CD$). Regular grammars are even more restricted. There can be only one symbol on the left and at most two symbols on the right, at least one of which must be a "terminal" symbol, that is a symbol which cannot appear on the left. In addition, in a regular grammar the terminal symbol must always be on the right or on the left ($A \rightarrow aB$ or $A \rightarrow Ba$ and $A \rightarrow b$, but not $A \rightarrow aB$ and $A \rightarrow Ba$, where lowercase letters denote terminal symbols). Almost all modern parsers are based on context-free or regular grammars (although they are often "augmented" with additional information). Because of their form, parses based on these kinds of grammars can be represented as trees (see figure 2).

Figure 2: Tree For Context—Free Analysis of "The man bit the dog."
Finally, parsers can be identified in terms of the linguistic grammar that they use or are based on. At the moment, the three Most important linguistic theories are Government-Binding theory (GB) which is most closely identified with Noam Chomsky (see Chomsky 1981); Lexical-Functional Grammar (LFG), which is particularly associated with Joan Bresnan (Bresnan 1982); and Generalized Phrase Structure Grammar (GPSG), which is associated particularly with Gerald Gazdar and Geoffrey Pullum (Gazdar et al. 1985). Although often mentioned in discussions of theoretical linguistics, transformational grammar is not, in fact, a theory which is being actively developed; it has been superseded by the three theories mentioned above, in particular by Government-Binding theory, which, of the three, is its most direct successor. For a variety of reasons, GB theory is not at this time the basis for many parsers, either within the domain of CALL or outside. Most parsers which use a linguistic theory are based on LFG or GPSG.

The various ways of classifying parsers relate directly to the ways in which they operate. If a parser is of the WASP variety, for example, it deliberately tries to find an acceptable parse without backtracking; if it is also based on GB grammar, it tries to create parses of the type produced in this linguistic theory. To understand the nature of a particular parser means, among other things, to understand the particular decisions which its developers have made in relation to these classifications.

The classifications discussed above can also help in understanding practical aspects of implementation. An IBM project called CRITIQUE provides a concrete example of this. CRITIQUE (formerly EPISTLE) is a writing aid which analyzes texts for grammar and style errors and works in conjunction with a standard text editor. CRITIQUE uses an augmented context-free parser (that is a context-free parser which includes additional information to help parse) and involves a bottom-up, parallel-parsing approach (Heidorn et al. 1982, 307-308). In tests, the system has shown itself able to handle the parsing of a variety of different types of sentences. However, the fact that CRITIQUE employs parallel parsing means that, even apart from the question of its size, it might not be easily transferred to microcomputers which, at least at this point, are simply not designed for any kind of parallel processing.
Before leaving the subject of parsers, let us look briefly at their applications. Most CALI discussions of parsers have focused on how they can be used for strictly grammatical instruction. Without doubt, parsers can be used in this way to serve an important instructional function. Automatic syntactic error detection can potentially free teachers from a significant part of the labor involved in marking student writing, leaving them with more time to interact with students. In addition, the focus of instructional parsing on error detection and correction may lead to CALI-AI making a substantial contribution to the general theory of natural language processing: the ability to detect and correct errors is an important part of the human ability to use language and should also be an important component of any theory of natural language processing.

However, parsers have another function which ultimately may be more important than those just discussed. Since the syntactic analysis of an utterance is generally considered preliminary to its semantic/pragmatic processing, a parser is probably a necessary component of any program which attempts to "understand" language. Parsers then are likely to play an important role in building CALI-AI systems which can truly replicate the linguistic behavior of a teacher.

**Semantics/Pragmatics**

Far less developed than syntax but, for language instruction, ultimately at least as important, is the formal understanding of meaning. Semantics/pragmatics can be divided into the study of word, sentence and text meanings.

Let us look first at the study of word meaning. Traditionally in semantics, words are described in terms of taxonomic classes. So, for example, the meaning of the word "cat" can be described in terms of the kinds of things we assume a cat to be: an animal, an animate entity, etc. These taxonomic aspects of the meaning of words are often encoded in terms of meaning postulates or features. A meaning postulate can informally be said to have the following form: "take what you will, if it is an $x$, then it is also a $y$." A meaning postulate for the word "cat" would thus be "take a cat, then it is also an animal." On the other hand, features are in effect the names of the taxonomic classes. Thus "animal" could be considered a feature of the word "cat." In terms of AI this kind of taxonomic relation is often described as being either an AKO (A KIND OF) or an IS-A (IS A). The former is used to describe the relation between classes; the latter, the relation between an individual and a class of which it can be considered a member. So, for example, the concept cat has the AKO relation to the concept animal, while a particular cat (or a word denoting one) would have the IS-A relation to the
concept cat (as well as implicitly to the concept animal). Such subordinate/superordinate relations (see figure 3) can clearly result in lengthy chains: *cat* is a subordinate of *mammal*, which is a subordinate of *animal*, which is a subordinator of *animal entity*, etc. (see Lyons 1977 and Winston 1984, 253-266).

![Figure 3: Tree Representation of Concepts Related by the Subordinate/Superordinate Relation](image)

There are, however, many aspects of meaning which cannot be described in terms of taxonomic classes. This is particularly true of what sentences express. Take, for example, a sentence such as "John robbed a bank." A large portion of the information expressed by this sentence is not taxonomic: it involves the relations between John, the act of robbing, and the bank. The problem is how to represent such meaning formally. Roger Schank has proposed the use of specified primitive relations for this purpose. These are used to describe meanings in terms of a *conceptual dependency network* (Grishman 1986, 101f). Among the primitive relations are the following:

- **ATRANS** - the transfer of an abstract relationship such as possession, ownership, or control.
- **PTRANS** - the transfer of a physical location of an object.
- **GRASP** - the grasping of an object by an actor.
- **SPEAK** - the action of producing sounds. (from Grishman 1986, 101)
Other methods of handling these aspects of meaning are available (see, for example, Bailin 1987). However, for the purposes of this article the issue is not what the optimal method is; what is important is that there is a problem.

Formal descriptions of semantic relations can be represented as "semantic networks." These networks are constituted of labeled nodes (circles) and labeled arcs (lines) which connect the nodes (see Grishman 1986, 95ff). As can be seen in figure 4, the nodes can represent predicates and entities and the arcs, the relationships between them.

![Figure 4: Semantic Net](image)

When we use sentences to say something, we are creating "texts." One major task of that part of AI known as computational linguistics is to understand text meaning. Nevertheless, very little progress has been made in devising software which can take any text and derive a set of inferences (that is, propositions which the text implies). On the other hand, a great deal of progress has been made in limited semantic domains involving specified contexts.

Of particular importance in this regard are schemas or frames (called scripts in relation to narrative structures). Schemas are sets of statements containing variables (see Charniak and McDermott, 405ff). In programs which use schemas for "comprehension," a text is broken down so that the variables in the schema are replaced by concrete information in the text. By way of illustration, we can use a simple example. A weather report will characteristically deal with a certain number of classes of information—temperature, barometric pressure, and wind velocity, for example. A program to understand the text of a weather report could contain a series of statements which would have variables standing for these items. The text then could be broken down in such a way as to
supply actual information in place of the variables. The program would not "understand" a weather report in the way in which people do, but it could still take from it important information about the weather.

Schemas can be enhanced with other semantic information to create a somewhat fuller interpretation. Imagine, for example, another schema, this time for a dog show. The statements which make up the schema have variables which can be filled in for the various types of dogs. In addition, the database for such a program contains information about each dog species. When a variable slot is filled in with the name of a particular breed, the program can also generate additional information, using the information in its database about that breed.

Schemas are clearly nothing more than a beginning. However, as of this point, they are still the major means of handling meaning within AI. More general procedures which are not as context bound await development. In particular, a "smart" semantic component should be able to take word meanings and, using combinatorial principles, put them together to derive sentence and text meanings. At a minimum, such a procedure should be able to generate a set of statements which a text implies. For example, take the sentence "The woman entered the house." A semantic component with combinatorial principles should be able to take this statement and come up with a set of statements which it implies: "There was a human who went into a house," "there was a woman in the house," etc. It should then be able to take these statements and relate them to the implications of the other sentences in the text—for example, the implications of the sentence "the woman was a burglar."

In addition, a "smart" component should be able to generate information which is implied but which, strictly speaking, is not part of the descriptive meaning of what is said. Such information results from what are called "speech acts" (Grishman 1986, 156-158, and Lyons 1977, 725ff). Assume you are sitting in a room with someone and the window is open. If the person says to you "It's cold in here," this can be taken to mean that that person is trying to get you to close the window. The inference does not come from the sentence itself, but rather from the utterance within a particular type of context. The challenge is to formalize the conditions under which we make such inferences.

The reader should not assume that the relative paucity of semantic/pragmatic work means that it is impossible to create a smart semantic component. At present we have no reason to assume that the properties discussed above cannot be formalized. Moreover, some interesting work has been done, particularly by the University of Delaware team of Culley, Mulford,
and Milbury-Steen (1986). This team has been actively engaged in the
development of intelligent CALI adventure games. The programs use scripts
which specify the kinds of knowledge needed at various points in the game and
"case frames" which identify the semantic nature of the grammatical arguments
of verbs (subjects, objects, objects of prepositional phrases, etc). So, for example,
if the French verb "entrer" is used by a student, the case frame stipulates that the
object of the prepositional phrase beginning with "dans" must be a place adjacent
to the one in which the student is "located" in the game. Frames are also used as
part of the database for the "world" of the game to specify the properties of
entities. So, for example, a specific table is described in the database in the
following way:

- substance: wood
- color: brown
- weight: 80 pounds. (Culley et al. 1986, 85)

Within this representation, the items "substance," "color," and "weight" identify
frame slots for which specific values are filled in. By using techniques such as
these, the developers hope to create software which can engage in dialogue and
"penalizes unremitting misuse of language and rewards lexical richness" (Culley
et al. 1986, 73).

This project focuses on everyday language and thus indicates the direction
which CALI-AI must take. Language instruction concentrates on non-technical
discourse. Unlike the front-end for a technical expert system (that is, the part of
the system with which the user interacts), CALI-AI must be able to handle the
"general" dialect rather than a specific scientific or technical "sublanguage" (see
Kittredge and Lehrberger 1982). In many ways this makes the challenge for
CALI-AI greater than for many front-ends, since it is well-known that the most
difficult parts of language to handle computationally are those which are the
least technical and consequently the least defined from a formal perspective.

**Morphology**

Although morphology is not often discussed as a separate component of
natural language processing, this may well be because in English it plays a role
secondary to syntax in determining grammatical relations. However, in highly
inflectional languages such as German it is at least as important in this regard; in
languages such as Latin, morphology rather than syntax plays the leading role.
The point is not merely theoretical. If one is trying to teach a language such as Latin, designing a syntactic parser is almost beside the point since within clauses word order is relatively free. More relevant to issues of grammaticality is a morphological parser which can identify various kinds of inflections and use the information to build the grammatical (and logical) relations between words. Moreover, in a language like German, the lack of a morphological parser can make it almost impossible to develop a successful syntactic parser.

Nevertheless, relatively few computational strategies exist for morphological parsing. Even a Successful CALI project such as the University of Delaware’s Latin Skills Program (see Culley 1984) uses simple enumeration in order to identify particular morphological forms. While this is quite successful as long as the task is limited, it is not an efficient approach.

On the other hand, CRITIQUE, which attempts a full parsing of open-ended texts, employs more sophisticated methods (Heidorn et al. 1982, 310ff). Because listing in a dictionary every possible derivational and inflectional form of every word would make the process of dictionary look-up extremely slow, CRITIQUE uses computational linguistic strategies to parse morphological forms and consequently simplify the look-up procedure. The strategies are based to some extent on the morphological theory of Mark Aronoff who claims that words formed by productive word-formation processes do not have to be listed in the dictionary (Aronoff 1976). So, for example, the IBM group notes that "prioritize" is not listed in their dictionary. Instead, they have a rule which stipulates that a noun ending in \textit{y} can be turned into a transitive verb by dropping the \textit{y} and adding \textit{-ize}. In a somewhat different vein, two template programs developed at The University of Western Ontario, VERBCON and COMTEXT, use a WASP-style approach to morphological parsing to handle the identification of English and French verb tenses respectively (Bailin and Thomson: forthcoming, and Holmes and Bailin, forthcoming). The approach allows these programs to identify verbs more quickly than they could otherwise. Nevertheless, despite such endeavors, the morphological arena is nowhere near as crowded as the syntactic one. However, the importance of morphology for languages such as German and Latin may help to spur the development of more sophisticated approaches.

Speech Processing

Although almost completely ignored in most general introductions to AI, speech processing may eventually become one of the most important components of CALI-AI systems. If systems can be developed which have the ability to produce appropriate speech models, as well as to process and correct
student's speech, the computer (with appropriate peripherals, of course) may be able to give students the practice speaking a language which cannot possibly be given in a classroom setting.

However, while the future may be bright for applications, it is not at all clear that the road to those applications is smooth. At the moment, speech processing technology is more a matter of signal processing than an application of AI strategies (that is, strategies which attempt to replicate the ways in which humans process language sounds). Perhaps the, biggest stumbling block is the fact that speech processing involves input from a number of different sources. An example will perhaps make the problem clearer. Assume that a speech signal contains the following sounds:

\[ k \ a \ t \ s \ k \ a \ r \ s \] (phonetically \( ka \ tsk yrz \))

This sequence of sounds could be interpreted as "cats cares" "cat's cares," or "cat scares." In order to decide which of these possible identifications is correct, syntactic, semantic, and even contextual information may be necessary. The HEARSAY project has developed the concept of a blackboard to handle this problem. The system contains a number of independent "knowledge sources," each of which contains information pertinent to a specific domain (for example, a syntactic or semantic domain). Pertinent hypotheses created by a knowledge source are written on the blackboard and through the blackboard are available to other knowledge sources. In this way the system can utilize the various kinds of information in order to make an identification of a sound sequence (see Rich 1983, 278-281 and Winograd 1983, 403).

Nevertheless, despite innovations such as the blackboard, the immediate future for speech processing in CALI is bleak. Far more work is needed before the technology is developed enough to have truly useful language teaching applications. In this regard, it should be pointed out that a truly complete speech processing component for CALI-AI would need more than just the ability to identify words. It would also need the ability to guess at the intended word when a student mispronounces it. Since mispronunciation and the ability to guess at what is intended (with a fair degree of success) are properties which human beings exhibit when using language, CALI-AI work may eventually make a definite contribution to the theory of natural language processing in this area.
Language Generation

Let us now turn briefly to language generation. In language generation, knowledge from various aspects of linguistics and natural language processing—syntax, semantics, morphology—is brought to bear upon the problem of having a machine produce utterances.

Random generation of sentences has been employed in some CALI-AI software. Henry Decker and Tom Rice’s VERBSTAR and Alan Bailin and Philip Thomson’s PARSER are examples of such programs. The former is intended to teach the use of French verbs in sentential contexts; the latter, the use of English sentence structure (Bailin and Thomson forthcoming). Although programs such as these can be quite valuable, it is generally assumed that the most important use of language generation will be in programs which "talk meaningfully" with students.

Researchers have developed some interesting programs which can generate meaningful discourse in limited domains (see Grishman 1986, 159-170). However, the technology has not as yet been applied to CALI-AI. What discourse programs do exist are of the ELIZA type (see Underwood 1987 for discussion of LIESL and FAMILIA, two CALI programs of this type). These programs do not understand what students are saying, but rather react to keywords. Say, for example, that a student types in a phrase such as "I am x," where x is some affective term such as "happy," or "sad." The program identifies the form "I am x" and responds with a canned response such as "why are you x?" Note that in giving this response, the program can in no way be said to have understood the utterance; rather it has simply identified the form of the utterance and responded by replacing a variable in a schema ("why are you x?") with the affective term. The effect can often be similar to the response of an understanding speaker, but only if the conversation takes the turns anticipated by the program’s designers.

Perhaps the major problem in applying more sophisticated language-generating techniques to CALI, is that language instruction, as already noted, generally pertains to everyday language rather than specialized discourse involving technical and scientific domains. Unlike technical terms, words in everyday language are not usually defined precisely. In addition, the syntax of everyday language is generally far more fluid than that of technical and scientific discourse. The rules which govern everyday language are thus more difficult to formalize in a way which a computer can handle. This difficulty, however, can here again be a challenge, a place where CALI-AI can make an important
contribution, since there are few other areas of practical application which necessitate such a general approach.

**Language Learning**

Could a computer learn the rules of human languages? At least in principle the answer is yes. The theoretical underpinning of such endeavors is the mathematical theory of learning (see Gold 1967 and Osherson et al. 1986). Mathematical learning theory attempts to articulate a formal account of learning, particularly of language learning.

Whether considered from the perspective of mathematical learning theory or from a more pragmatic viewpoint, there appear to be at least two prerequisites for language learning (Osherson et al. 1986, 7ff and Winston 1984, 39ff). First of all, a learner (be it human or machine) needs samples, that is data from which to learn the language. Second, the learner needs some conjectures concerning the grammar which are at least in part based on the samples. In addition, the learner may need, at least at points in the learning process, indications of whether or not particular samples are in the language (Osherson et al. 1986, 113ff). It might be noted that those which are part of the language are called "positive"; those not included are called "negative" (Winston 1984, 391).

There are many unanswered questions about the nature of language acquisition. One of the most important in relation to human language acquisition is how much knowledge of language is "built-in" (that is, part our biological makeup) and how much actually learned. In a similar vein, one of the most important questions in relation to machine language learning is how much linguistic knowledge must be part of the program (or the machine itself) and how much learned from samples. Until we know far more about what knowledge is needed to learn a language, we will be unable to build programs which can replicate the human acquisition process.

Despite our relative ignorance about the nature of the acquisition process, a number of projects have attempted to construct language-learning programs (see, for example, Brand 1987 and Winston 1984, 416ff). However, as far as I am aware, these projects have not led to any CALI applications up to this point. Nevertheless, automated language learning may ultimately have much to contribute to CALI. Perhaps the most important application would be to allow us to develop a precise model of a student's grammar at a particular point in the student's development. Comparisons could then be made between the "target" grammar of the language and the student's. On the basis of the comparison,
Particular gaps in the student's grammar could be identified and various kinds of remedial action taken.

**Problem Solving**

We often think of problem solving in relation to expert systems designed for military or industrial technical applications. Nevertheless, as I hope to show in the next section, it will in the future be of crucial importance to AI applications in CALI. However, before looking at what it can do in CALI, we should have an idea of what it is.

Two basic strategies for approaching problem solving within AI are the "generate-and-test" approach and the "rule-based" approach (Winston 1984, 163ff and Rich 1983, 73f and 31ff). Let us look at each in turn.

The "generate-and-test" approach, as its name implies, involves solving a problem by producing plausible solutions and testing to see which of them are appropriate. The generator may produce all possible solutions before the tester takes over, or the generator and the tester may alternate. If, for example, the problem were to decide the best way to tell a student to make corrections in an essay, the "generator" could produce all the plausible ways of telling the student before the tester evaluated them to see which was appropriate (in terms of diction, style, etc.). On the other hand, the generator and tester could alternate, producing and testing possible ways of telling a student until an appropriate way was found.

The "rule-based" approach operates in a somewhat different manner. In this approach a set of rules is applied to a problem. These rules are generally of an if-then variety. The simplest form of such an approach is a "situation-action" system. If certain conditions are met, then certain actions must follow. A system of this kind is considered "deductive" if the result of a condition being met is a new fact rather than an action. A deductive rule-based system can be "forward chaining" or "backward chaining." A forward-chaining system is one which moves from facts to conclusions. A backward-chaining system is one which works in the opposite direction—from conclusions to the conditions from which they resulted.

An example may help to make the concept of a rule-based system somewhat more concrete. Let us say again that the problem is to find the best way to tell a student to make corrections in an essay. Our rule-based system determines the best way on the basis of certain attributes of the student. If, for example, the student is a young freshman then statement A, "Please check your reference book carefully to find grammatical errors in this passage," is considered appropriate. If the student is an older graduate student, statement B, "Please use
all available resources to check for errors,” is considered the best way to tell the student. Our rule-based system first "reviews" the facts about the student to see if they match one or another set of conditions. If the facts match the first set of conditions—that is, the student is a young freshman—then the system indicates that statement $A$ should be used. If, on the other hand, the student is an older graduate student, then the facts match the second set of conditions given above and the system indicates that statement $B$ should be used.

Problem solving is not a distinct component of intelligent CALI systems at the moment. It is, however, likely to become one as such systems become more sophisticated. As CALI-AI programs become increasingly complex, issues to which problem-solving techniques can apply will have to be treated in a deliberate manner, and specific problem-solving routines developed to handle them. Of the possible applications of problem-solving theory in CALI-AI, the most important may well be in modeling teacher behavior, particularly in simulating the way a teacher decides on the best teaching strategy to use for particular students in particular situations.

**Modeling Behaviour**

Modeling is the attempt to produce on a machine a simulation of human behavior. Although the issue of modeling is not one which is, at this time, much discussed in relation to CALI-AI, it is crucial to the field. As was noted at the beginning of this article, the ultimate goal of CALI-AI is the machine replication of human language-teaching behavior. Thus, successes in the modeling of language-teaching behavior are successes in achieving the main goal of CALI-AI.

It might seem natural to discuss modeling in terms of one or another psychological theory of learning. Yet it must be kept in mind that in relation to language acquisition, particularly second language acquisition, we know rather little and a great deal of what we do know derives not only from psychologists, but also from various subbranches of linguistics (applied linguistics, psycho- and sociolinguistics, etc.). Indeed, language learning may well turn out to involve a distinct kind of learning which cannot be easily subsumed as simply an instance of general learning theory—and certainly cannot be subsumed at this point, given the present state of such theory.

The real issue for CALI-AI is not the question of which teaching and/or learning method is the best. It is unlikely that AI applications to CALI can settle this issue any more definitively than we have been able to without the technology. Teachers, in fact, use a whole range of approaches. Some prefer to
use Krashen-type communicative approaches, while others prefer more "traditional" grammar and vocabulary work; still others prefer a combination of strategies (see Richards and Schmidt 1983 for discussion of various approaches). Successful CALI-AI must be able to replicate such strategies, not choose among them.

What, then, are the qualities which must be replicated? First of all, CALI-AI must have the ability to present stimuli in ways in which a teacher would: in the form of drills, in the form of conversation within or independently of a simulated context, etc. Second, it must have the ability to process the student's responses to the stimuli and, in so doing, correct errors as a teacher would. Finally, it should have the ability to give hints, supply correct forms and teach vocabulary, much as a teacher would, in relation to the student's ability and the kind of learning situation (tutorial, simulation, etc.).

These qualities cannot be actualized simply by creating an independent component of CALI-AI systems which "models" language-teaching behavior because, to a large degree, the modeling of language-teaching behavior is a function of the "technical" components of a CALI-AI system. It is a function of the natural language processing capabilities insofar as these are necessary for creating the learning environment, for processing responses, and for giving corrections and hints. It is a function of the problem-solving capacities insofar as the system—like a human teacher—must make choices about the kinds of correction to be presented to students and the type of remedial instruction to be offered. It is a function of the machine language-learning capabilities of the system insofar as it is necessary to simulate a teacher's ability to follow a student's progress and to imagine how the language looks to the student at various stages of the process.

Perhaps the most notable endeavor in modeling at this point has been the PARNASSUS project at Carnegie Mellon University where Neuwirth et al. have been engaged in trying to apply Anderson's ACT* theory of skill acquisition to teaching students to write effective sentences in English (Neuwirth 1986). According to Neuwirth, ACT* theory holds that people acquire skills by learning to apply "declarative" ("textbook") knowledge to situations through practice. The result of the learning process is procedural" knowledge, that is knowledge of how to do something. PARNASSUS uses sentence combining to allow students to acquire greater ability to write more effective sentences. After the student makes a revision, PARNASSUS evaluates it, and if there is a better revision, it tells the student this and asks the student to try to create it. What is perhaps most interesting about this project is that the researchers are asking themselves in a
self-conscious manner what the most effective way of teaching would be and thus what behavior the program should be imitating. Should PARNASSUS (as it does at present) simply tell the student that there is a better way of revising or should it use various kinds of examples? Our ability to model language teaching behavior can only be increased by such deliberate efforts to understand its nature.

Conclusion: The Role of Language Teachers

For those of us interested in the technicalities of creating CALI-AI software, perhaps the most vital contribution we can make is not to the actual programming but rather to the creation of algorithms, that is, the step-by-step procedures which are the basis for the code. Language teachers have not only an intimate knowledge of the target language but also expertise in the strategies for teaching it. We are, then, in a particularly good position to contribute to breaking down both the language and the instructional process into the step-by-step procedures necessary for creating CALI-AI.

There are already a number of projects in which language teachers are contributing in this manner. For example, Ruth and Alton Sanders have been developing a German essay processor called SYNCHECK, which is intended as a writing aid for intermediate and advanced English-speaking students of German (Sanders and Sanders 1987). Ruth Sanders, a German specialist, has the primary responsibility for producing the grammar and instructional procedures which are used in the program. Alton Sanders, a computer scientist, has the primary responsibility for the technical aspects of the programming.

Those who have neither the time nor the inclination to become involved in the mechanics can still help to answer many important questions. Are there kinds of feedback which should be suppressed in CALI-AI to avoid information overload? For interactive courseware, are there particular types of conversational situations which might be especially useful? Language teachers must answer such questions if software is to be developed which suits the needs of their students. And they must become familiar enough with the technology to express these needs in an effective way.

I would like to thank Glyn Holmes (editor of Computers and the Humanities) and Robert Mercer (Department of Computer Science, The University of Western Ontario) for their helpful criticisms and suggestions.
Notes

1 I could have used the term "ICALI" (intelligent computer-assisted language instruction). However, it implies that current software of this type is intelligent, an implication I prefer to avoid.

References


**Author's Biodata**

Dr. Alan Bailin is an English Usage Specialist at The University of Western Ontario. He is the chairman of the CALICO AI SIG. At present he is co-editing a special issue on artificial intelligence and computer-assisted language instruction for Computers and the Humanities.

**Author's Address**

Alan Bailin
Effective Writing Program
University of Western Ontario
London, Ontario
Canada N6A 3K7