THE COMMA CONVERSER: AN INTELLIGENT COMPUTER-ASSISTED LEARNING PROGRAM TO TEACH THE USE OF THE COMMA

Arnold Keller

A Thesis

in

The Faculty

of

Arts and Sciences

Presented in Partial Fulfillment of the Requirements
for the degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

MARCH 1982

ABSTRACT

THE COMMA CONVERSER: AN INTELLIGENT COMPUTER-ASSISTED PROGRAM TO TEACH THE USE OF THE COMMA

Arnold Keller, Ph.D.

Concordia University, 1982

Using commas properly requires a serious learning effort for most college-level students, and there have been several attempts to teach it through Computer-Assisted Learning (CAL). But conventional frame-based CAL depends on sentences that have been programmed into the computer in advance instead of sentences that a student might himself write. This means the resulting lessons are exercises in recognizing errors in existing sentences, rather than in the composition of ones which are error-free. The thesis of this work is to show that a CAL program not subject to these limitations can be built and run by using an algorithmic representation of knowledge, a formal model of the student, and a metalanguage.

The COMMA CONVERSER (or COMCON) overcomes such limitations by accepting as input any sentence the student enters, creating an internal representation of it, and

initiating a set of exchanges with him about the correctness of the sentence's commas. This internal representation is based on a pattern-directed inference system which scans the input for key sentence features in configurations which indicate a high likelihood that commas should be used or omitted. These features have been identified by means of a new model of a properly-punctuated sentence, based on observed error behavior. In addition, COMCON builds a model of the student's input sentence that shows the presence and/or absence of these features, and as the exchanges with the student proceed, this model is updated to include data on the student's syntactical and semantic perceptions of his sentence. These data permit more precise instruction by COMCON. The exchanges themselves are in the form of "conversations" in which agreement is sought between the student and COMCON about the use of commas with the following: coordinating and subordinating conjunctions, nonessential and introductory elements, conjunctive adverbs, and about the elimination of extraneous commas. In addition, there is a HELP facility which can be accessed at any time by the student and which offers both information and practice on concepts encountered during the exchanges. A recapitulation of the session and some brief remediation is given at the end of the set of exchanges.

COMCON is rather expensive to run because its logic requires it to scan the input sentence many times and its size requires much memory (approximately 350K of an APLSV system using an IBM 370). This means widespread implementation must await the advent of sufficiently powerful yet cheap stand-alone computers.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

AN OVERVIEW OF THE THESIS

This thesis describes and tests the feasibility of the COMMA CONVERSER (or COMCON), an intelligent CAL program that takes as input any sentence the student writes and enters into a conversation about the correctness of its commas. Chapters III through VI present the program's design and deal with three main activities: building an internal representation of a student's input sentence, modeling his comma use, and designing the exchanges between student and system. To start, however, Chapter I outlines the problems involved in using the computer to teach English and surveys a number of existing programs. Chapter II is a critique that raises a number of practical and theoretical objections to frame-based CAL. Chapter III begins the part of the thesis which proposes the solution to the problems of frame-based CAL in teaching writing skills. It starts by discussing the various ways one might represent the knowledge needed to use commas properly and then offers reasons why the production-rule formalism is best-suited to this application. Chapter IV describes SENTCHECK, the part of COMCOM that takes the student's sentence and creates the internal representation COMCON uses. Chapter V reviews some attempts to model the student in existing intelligent CAL and presents the matrix model used by COMCON to measure the differences between the student and its own "master performer". Chapter VI describes the sorts of conversations open to a system that has gathered information of this sort about sentence and student. It begins with an overview of the program and a number of sample runs of COMCON before examining its logic in greater detail. It also discusses some practical considerations of implementation and COMCON's "help" facility. Chapter VII presents the main conclusions of the study and discusses appropriate future research. Apprendix I provides details of the model of a properly-punctuated sentence. This is the model on which COMCON is based. Appendix II reports a formative evaluation of COMCON's instructional effectiveness. Finally, I follow Boden's (1977) elegant solution to the problem of sexist language: let the author's own gender determine pronominal reference.

ACKNOWLEDGEMENTS

I have incurred a number of debts in the writing of this thesis, and it is a pleasure to acknowledge them here. Richard Schmid and David Mitchell both read portions of the thesis in its earlier stages. Gary Boyd, my supervisor, gave generously of his time and comments. He has helped me make this a better thesis than it was. Gordon Pask offered encouragement and direction at an important juncture. Gary Bourgeois did the final draft of the illustrations. And, of course, my students at Vanier College patiently lived through many false starts and many updated versions of the program. To all, my sincere gratitude.

A.K.

March, 1982.

CHAPTER I:
AN INTRODUCTION TO THE DIFFICULTIES IN USING THE COMPUTER TO TEACH WRITING SKILLS

It is a melancholy observation among teachers of writing that the comma is not so often used as misused. Why writing skills in general are poor has been well-aired to the point of commonplace: an overall decline in literacy throughout the culture, the abdication of schools in the teaching of basics, the impact of television, and so forth. Whatever the causes, the unhappy fact is that few college or university students can write clear and coherent sentences. Using commas properly will not instantly produce master writers, of course, since too many other factors are at work. But because the comma is a logical marker that shows relationships among the constituent parts of a sentence, its correct use can do much to clarify meaning. Post-secondary schools have recognized that remedial courses in composition must be given, but in times of budget cutting, the requisite small enrollments for these courses seem increasingly threatened. At first blush, the computer appears like an ideal solution: it is well-suited to the repetitive drill of fundamentals, and its cost has dropped precipitously over the past twenty years. And indeed, there have a number of attempts to develop courseware to teach the comma and other components of the freshman English syllabus. Unfortunately, neither the products nor the results have been outstanding.

Research has shown that Computer-Assisted Learning (CAL) can be successful in certain carefully delimited areas and uses. A number of research reviews report CAL as an effective supplement to traditional instruction (a notoriously fuzzy term). For example, Edwards, Norton, Weiss, and Van Dusseldorp (1975) claimed that CAL was in general equal to or better than classroom instruction in eight college-level projects. Their survey also found a saving of time in learning (a finding the literature often gives) and that low-ability students benefited most. Thomas (1979) found good results for CAL in achievement, retention, and especially in shortened learning time. Hausmann (1979) basically agreed with the findings of the other two studies, although he spoke somewhat more of CAL's "potential", particularly in the tutorial and drill-and-practice modes rather than problem-solving, gaming, and simulation. He too reported that CAL's self-pacing produced more rapid learning, again for lower ability students. He did not, however, concur about retention rates, seeing them as lower than in traditional approaches. He also stressed the need for longitudinal studies to determine if "expressed enthusiasm is due to the novelty of the CAI mode".

Kearsley (1977) dealt with the conceptual issues of CAL, arguing that theoretical problems were greater obstacles to success than economic or technical ones. He worried that the "premature extinction of CAI" was a real possibility unless research began to confront notions of individualized instruction, meaningful assessment of CAL's effectiveness, and possible interactions with cognitive psychology and artificial intelligence. While granting CAL's potential, Kearsley still argued that "it will be necessary to engage in considerably more theoretical work and research on conceptual issues" than had been the case in the past (p. 15). Although the relative brevity of his paper did not permit in depth analysis of many programs, Kearsley's comments are entirely pertinent to most existing CAL.

A far more comprehensive review of CAL is Rockart and Morton's report to the Carnegie Commission (1975). They survey in some depth such major CAL efforts as Plato, TICCIT, the Stanford work in mathematics and logic, the Florida State programs in physics, and many other projects. They delineate the major approaches to CAL--tutorial, drill-and-practice, dialogue, gaming, and so forth--noting at the time of writing that although only sixty-six percent of programs fell into the tutorial or drill-and-practice category, ninety- five percent of the money had gone there. What had been the result of such expenditure? They come to the following position:

"In summary, the results in the

drill-and-practice and tutorial modes of instruction are ambiguous at the university level. It does appear that there is evidence to support their use for structured, repetitive kinds of material; . . . possible misapplication of the technique vis-a-vis substantive course material, plus the general failure to do research on the impact of such systems, has resulted in a feeling of disenchantment. Little new work except in remedial learning is reported with regard to initial presentation of the material, and no supportive evidence is available in the literature" (p. 104).

The literature, therefore, is difficult to interpret, especially since (as Hausmann and also Rockart & Morton report) there is difficulty finding well-controlled and sound research. Yet there is evidence, though often anecdotal, that CAL is effective in motivating students, in decreasing the amount of time spent meeting objectives, and, in the case of highly structured and mechanical material, in producing achievement. What the literature often fails to consider or fails to demonstrate is a transference of skills from the narrow objectives of a CAL lesson to the broader contexts of learning. Transference of learning is important since in CAL one presumably uses the computer as a means to some further instructional end. Moreover, since existing programs generally work with basic skills, one would wish to see what Gagne (1965) calls "vertical transference", that is, the learner being able to do more complex things as a result of previous instruction. This issue is discussed more thoroughly below.

The literature that bears directly on teaching writing skills is even more difficult to assess, both because the of the dearth of controlled studies and the relatively few programs that have been written. On this last point, the data on simply the number and kinds of CAL programs that have been written and implemented show comparatively few that deal with language.

As part of their report, Rockart and Morton surveyed all university institutions in the state of Massachusetts in 1974 and found that no institution reported any use at all of CAL in their English curricula at either the undergraduate or graduate levels. And this despite very considerable use in other disciplines. Rockart and Morton also show a 1966-67 Southern Regional Education Board (SREB) in which only 0.1 percent of students nationwide used CAL in English courses. These figures are not surprising, of course, because English departments are likely to be far more resistant to using computers than disciplines in either the natural or social sciences. Part of that resistance is often explained by the "humanistic" nature of English, but a more reasonable explanation is the recalcitrance of the subject matter itself. That is, language and literature are simply far less amenable to representation on a computer than mathematics or physics. We will return to this key issue, but for the moment, the emphasis is on the relative scarcity of CAL in the English curriculum.

The work that has been done takes the form of branching programs* that ask the student to respond to stored sentences and then route him to appropriate frames. Chapter II will discuss the theoretical objections to this strategy, but for now, it would be useful to review some projects which indicate the range of work that has been done in frame-based CAL to teach English writing skills. The first such project is quite typical of the simple kinds of branching programs one can write either with an authoring language or with BASIC. The others, while still depending on canned sentences, are somewhat more sophisticated in programming and overall design.

----
* "Branching" here means the sequence of frames to which a student is directed and which are contingent upon the responses he makes. See Crowder (1964) for the classic statement on "intrinsic programming".

Kline (1977) developed a set of modules intended for drill in an English as a Second Language course, but many of his topics are the same as those found in regular freshman or remedial courses (for example, sentence patterns, nouns, verbs, phrases, clauses, and punctuation). He intended the lessons to reinforce topics taught in class, and he scheduled an hour on the computer each day immediately after class. The instructional strategy used began with an exposition of principles accompanied by appropriate examples. This was followed by a series of multiple choice questions in which the student had to select either the one correct answer, the one wrong entry, or the sentence with more than one mistake. Correct or incorrect, the student received a statement of the relevant principle involved; this functioned as an instructional hedge against students simply guessing.. For instance, regardless of having chosen "Smith" or "Smith's", the student gets the following: "Both entries are proper nouns in listing number 2 and neither citational form ends with a sibilant sound.". Such output is quite typical of the program which presents relatively large amounts of exposition followed by multiple choice questions. Kline was not able to establish a controlled experiment but still claimed a number of benefits including a reduction of instructor time spent on review, enthusiastic student response, and easier detection of student problems. He did stress that although the CAL modules were effective as devices for reinforcement of materials taught in the class sessions, they were of diminishing effectiveness when utilized as self-instructional modules.

Thames (1972) produced a set of programs for remedial students covering such topics as diction, spelling, topic sentences, paragraph unity, sentence order, and so on. Written in APL, the programs use prepared sentences and paragraphs but often avoid the typical multiple-choice format by having the student change the words in a given sentence. For example, a student can transform the compound sentence "He doesn't like her, but I do" into a complex one ("Even though he doesn't like her, I do") by instructing the computer to "DROP BUT" and "INSERT EVEN THOUGH BEFORE HE". A few other programs allow similar manipulation of text.* Other programs present sentences along with one or two modifiers which the student can place after any word in the sentence. Since only one or two locations are permissible, the program matches the relevant portions of the new sentence against one or two prestored strings. There are also simpler strategies to evaluate responses. The spelling program prints out a sentence with a misspelled word and asks the student to find it. A program called SPECIFY asks the student to enter a "G" if a given sentence is a generalization and "S" if it is specific. A lesson on sequencing of sentences within a given paragraph requires only a list of numbers to be entered.

Although Thames' overall strategy avoids a simple multiple-choice format, the student still is restricted since he cannot legally enter a word which does not appear

----

* Thames does not specify the programming, but this is clearly the result of APL's power in breaking up and rearranging vectors of character strings.

in the program's prepared sentence. The same is true for the lessons which deal with paragraphs: the student cannot enter his own topic or supporting sentences but must choose from among those offered. Nonetheless, the design of these programs is quite good given the basic limitations in analysing language, a deficiency to which Thames herself calls attention although she rather naively sees this as a storage problem and as a lack of "special routines" to define grammatical rules. It is precisely the lack of such procedures that is at the root of the problem of developing good CAL English lessons, and not at all a matter of importing "compiler approaches developed for scientific applications" (p. 309).

Thames' project is not a controlled study, but she does provide a number of comments from students. Generally, the response is quite favorable, especially since her population consisted of students who had failed English entrance tests and had to take her course. She too noted that the lower the ability of the student, the more useful was the CAL.

Perhaps the most ambitious of all CAL work in English has been done on TICCIT (Merrill, Schneider, and Fletcher, 1980). (TICCIT is an acronym for Time-shared, Interactive Computer-Controlled, Information Television.) TICCIT includes work both in hardware and courseware, and claims three advantages over other CAL:

1) instructional tactics are built in, not created by individual authors;

2) the strategies teach concept-classification and rule-using objectives rather than memory drill or problem-solving;

3) the learner, not the system, maintains control over the sequence of instruction. TICCIT assumes that any instructional presentation consists of "primary presentation forms" (rule, example, or practice displays) and their elaborations. Each segment of a TICCIT program teaches either a concept, a procedure, or a rule by means of a wide number of display files which the student can select with the aid of various advising devices which show

prerequisites for an objective, descriptions of materials, and sample test items.
It is the courseware design that makes TICCIT, in the minds of its designers, superior to other CAL modes.

The English course that demonstrates TICCIT's design includes a section on pronoun-agreement that includes rule displays such as:

"A pronoun agrees in number with its REFERENT. Singular referents take singular pronouns. Plural referents take plural pronouns. Singular referents which have no sex indicated take the generic pronouns him/he/his/".

This is followed by an EXAMPLE display (with colour highlights) and then a PRACTICE display which asks the student to "Edit any pronoun in the passage below that doesn't agree in number with its referent." The student types his answer (or sometimes moves the cursor) and then receives appropriate feedback ("Correct" or "Wait. One of the words you changed wasn't even a pronoun.") In some other cases, feedback simply provides the right answer, much like an EXAMPLE display. A HELP mode is also available to provide additional information such as mnemonic devices, algorithms, heuristics, or various kinds of graphics.

This outline of TICCIT's strategy holds true for other topics that are covered: TICCIT's design specifies the consistency of its instructional logic. (Merrill et al. make this point explicit.) The system provides a great deal of exposition, very nicely presented graphics, but relatively little for the student to enter other than a single word. The student's chief activity lies in his selection of the next portion of instruction. This is quite different from having the system determine what to present next. Indeed, the designers specifically reject the "dream of a maximally adaptive system" which could assess a student's learning style, aptitudes, past achievement, and readiness. They argue that "A totally adaptive system, if it could be developed, would be maladaptive, making students . . . "spoon fed" (p. 10). The natural environment, they argue, is not as adaptive and a "horrible nightmare" of overly-dependent people might result.

According to its designers, TICCIT helps students become system-independent: "When a student learns to select the next best display on the system, he or she is also learning how to select the next best display in non-computer based situations" (p.10). This implies, of course, a transference of skills from the immediate teaching context to some other. But evaluations of TICCIT don't bear this out. Although TICCIT users at Brigham Young averaged 87.9 on a final exam (not significantly different from the 87.3 for other groups), the essay averages were no better than C-. The essay grade was the same for nonusers of TICCIT at BYU, although on a five-point scale, TICCIT papers were 0.24 better than non-TICCIT papers, significant at the .05 level. (But my own experience grading essays tells me that this sort of scale is much too fuzzy to be used in evaluations like the one here.) Whatever the test scores show, the claimed independence still produced results which "leave lots of room for improvement in both groups (p. 124) and which show that "performance isn't as outstanding as the developers had hoped" (p. 126). Not surprisingly, at the end of the section on TICCIT's evaluation, the authors change their notion about student "spoon-feeding": "The model needs to be expanded to handle the evaluation and advisement of students with different learning abilities, levels of experience with the course content, learning styles, and levels of motivation" (p. 126).

The CAL program this thesis describes begins with the dissatisfaction I felt about my own research into CAL in English. Keller (1979) described an experiment which measured the performance of students who had been taught the same course in writing mechanics by one of three different delivery modes: a printed linear textbook, a printed scrambled textbook, and a suite of CAL lessons. Some 100 students used material that taught such topics as dangling constructions, capitalization, the use of verbs, the case and agreement of pronouns, and most extensively, the use of the comma. About ten to twelve hours of instruction were produced. Typically, a lesson began with a statement of objectives, gave some instruction (often in the form of a model), showed how this instruction might be applied, and then gave the student a set of sentences to be corrected. Generally, a student had to select the word after which he might add or delete a comma or else make other changes. (The multiple-choice format was avoided as much as possible, although not always.) This strategy held for each concept taught (for example, "a comma is wrong if it precedes a coordinating conjunction joining two equal elements"), and there were about three such concepts per lesson. At the end of the lesson, the student wrote a 20-item criterion test. As with the three programs described above, a wrong answer caused the student to be routed to a remedial frame or directed him to a particular page in the printed, branching text. No significant differences were found among post-test scores, demonstrating that simply delivering a course via a computer made no difference in achievement.

The following year, modifications were made to the basic material and extra drill-and-practice (D&P) frames added. The research intended to test the hypothesis that adding drill-and-practice affected achievement. One group of students received instruction virtually identical to the previous year's. A second group, however, received the same lessons with the following enhancements: after practice on each concept, a student's score was calculated, and if he had not shown 100% mastery, he was automatically branched to a D&P sequence before being returned to the main stream of instruction. These D&P sequences contained about three to five extra questions and some further exposition.

However, the post-test scores showed that although both kinds of instruction were effective (at least in terms of what was being measured), there were no significant statistical differences between the two groups. The most the results indicated was that the effectiveness was apparent at a higher confidence level for the D&P group. Clearly, merely adding more drill and practice or more remediation without regard to the nature of that help was of marginal benefit.

Moreover, a disturbing fact became more and more evident. Although students scored very well on each lesson's criterion test and somewhat less well on post tests, examination of essays did not show improvement commensurate with test scores. This lack of improvement was in the very area that supposedly was being taught, and not in the more difficult area of style. And further, the results were observed over several sections and several teachers; the populations for both experiments were randomly selected from several remedial sections. Since the main objective from the very outset had been to improve essay grades, the results were not encouraging at all.

During the next year and a half, the problem of transference was confronted. Conversations with students revealed that the high scores on the criterion tests which followed each lesson were the result of the student looking for a particular kind of error and solving for that one kind exclusively. For instance, the lesson on using the comma with main clauses tested only for that skill; all the student had to do was determine which main clauses were present, check their punctuation, and make the correction. Indeed, the tests were not so much tests as drills. At the same time, this explained why the post-test scores, although showing improvement over pre-test scores, were not as good as individual criterion-test scores: On the post- test, many sorts of errors were present. A student could not simply look for a particular error pattern--main clauses and commas, for example--and make corrections. He would have to call upon his entire repertoire of skills to effect the necessary transference, but this he had not been fully trained to do by lessons which taught and tested one skill at a time.

The literature on transference--the ability to use what one has learned in a different situation--is varied and often in disagreement. For instance, it may be that specifics are transferred as is claimed in Thorndike's (1913) identical-elements theory; it may be principles as in Judd's generalization theory (1908); or it may be entire patterns and relationships as in various transposition theories (for example, Wertheimer (1959)). However, as Klausmeier and Ripple (1971) put it, there is this common ground: "the individual must perceive the new situation as being similar to that in which the initial learning occurred" (p. 618). And although one can argue that the concept of "the comma between two main clauses" remains constant whether one writes essays or takes a multiple-choice quiz, it does not follow that a learner will immediately recognize the similarity between contexts. Indeed, to use Gagne's distinction again, going from an example on a frame-based CAL lesson to the criterion or even post test is a "lateral" transfer because the tasks are both similar and at the same level of complexity. But going from the same CAL lesson to essay writing is a "vertical" transfer in that one must produce more advanced or complex outcomes. It follows, then, that effective CAL must provide practice in varied situations and not leave transference to the student's good fortune.

Garrison and Magoon (1972) review a number of the ways in which teaching affects transference. These include motivatating students, incorporating new materials into past learning, distributing practice and reviews, and generalizing the knowledge and skills being learned. They also stress the need to provide for application in a variety of situations. They conclude that

> "materials taught in a rote manner or as a process of memorizing facts or theorems will have little transfer value to the application of such facts and theorems to other situations, even though there may be some identical elements present. However, exercise in process and analysis does seem to improve the learner's ability to apply the exercised process" (p. 242).

This suggests that CAL, like other sorts of instruction, cannot cause transfer to take place unless it provides ways for teaching process as well as memorization. There may be, it is true, occasions when one simply wants a set of discrete facts learned, but this is not the case in teaching writing.

Gagne and Briggs (1974) also stress "setting some variety of new tasks for the learner--tasks which require the application of what has been learned in situations that differ substantially from those used for the learning itself" (p. 132). The example they cite happens to be the teaching of verb and pronoun agreement. Achieving transfer here, they argue, requires more than merely taking a sentence and varying the pronoun and the verb (which is a typical CAL strategy). They suggest asking the student to compose his own sentences with pronouns and verbs, rather than having him respond to the teacher's. Obviously, a live tutor can manage this much more easily than a computer; but the difficulties involved in CAL authoring do not obviate the need to do so if one expects transference to take place.

To return to my own research, the following year, "links" were added between the lessons to provide a context closer to that of essay-writing. A link tested not only the knowledge that was learned on the lesson immediately preceding, but the knowledge of all previous lessons. This was done by a series of sentences, each containing three separate sorts of errors. The student would be given the sentence and asked to enter a corrected version. Hence, the test required set of responses to a variety of problems rather than a single answer to an expected class of problem. To help the student, the program also taught the "IDT" algorithm (Keller, 1980) which showed him how to "identify" (I) places where errors might occur (such as near coordiniating conjunctions); how to "discriminate" (D) between what he saw and a model of correctness he had been taught; and finally, to "transform" (T) the sentence if the discrimination revealed an error. If these additional questions did not meet the all the criteria for transference one might find in the literature, they did provide more complex practice than before in processing language since the student had to analyze the sentences from several perspectives. They also offered some of the "variety" Gagne and Briggs specified. They did not, of course, take into account the truly important variety in the system: the student's cognitive processes could not be matched by the limited number of answers which the program could evaluate.

Moreover, there were numerous programming problems. Depending on whether or not the student found one, two, or three errors, the program branched him to different remediation frames. Since each link had about ten to twelve questions, and each question had three errors, some thirty to thirty-six samples of student behavior had to be considered. But for each of the three errors, there had to be separate frames for different response groupings. One could not give the same advice to student who got nothing right as to the one who missed just one. This led to rather elaborate programs, to say the least, which quickly faced inevitable combinatorial explosions. Implementation was beset by all sorts of debugging before the lessons were used in class.

At the same time, several new features were incorporated. In an effort to build up a sense of what the student knew, he was asked why he made the particular choices he did. Since this occurred each time he made an error, and since this information was rather cumbersome to collect, the process itself quickly became an obstacle. Indeed, student frustration at being asked repeatedly why he made an error led to the whole project being abandoned. A second attempted enhancement was the incorporation of a punctuation algorithm into the program itself. Although the algorithm had been shown effective in the classroom, its inclusion added to the complexity of programming. One had to make the algorithm available in different formats, depending on how far the student had gone in the lessons. Again, this made for some elaborate programming which took much debugging and which, indeed, a number of students found intrusive.

However, despite the presence of the student model, the algorithm, and above all, the extra questions, students did not improve their essay scores. They were simply unable to transfer the skills the individual lessons taught--and seemed to teach quite well judging by criterion tests. Five important considerations did, however, emerge from this work:

1) simply adding material to a branching lesson was not going to produce significant increases in achievement;

2) there was an obvious need to know more about the student than his test scores, but the information had to be gathered unobstrusively;

3) although an algorithm could produce good results, the actual teaching of it was probably best left off the computer because of the complexity of its programming.

4) a far more formal model of the properly-punctuated sentence had to be forthcoming if the computer were to have the necessary intelligence to teach;

5) most importantly, what the student was learning was not particularly helpful to him in writing essays, where it mattered most. Whatever skill was being imparted, it was not that of composing error-free sentences.

The research on using CAL for English writing skills, therefore, shows a distinct lack of success which Chapter II will deal with from a theoretical perspective. What the research makes clear, however is that using simple branching strategies in tutorials or using extra drill-and-practice does not demonstrably produce students who write better essays. Indeed, the usual research finding that low-ability students benefit most from CAI might be explained by their needs being so great that they can be met by even the most relatively primitive instruction. When better students and more complex skills are involved, much more is needed.

CHAPTER II:
A CRITIQUE OF FRAME-BASED CAL

Green ideas sleep furiously, Chomsky told us several years ago, and no green idea sleeps more furiously than the attempt to teach language by computer. On one hand, the promise of the computer to revolutionize instruction remains compelling; workers in artificial intelligence, cognitive psychology, and natural language manipulation forever seem to be at the very brink of a major breakthrough that will dramatically change education. But the realization of their visions remains remarkably recalcitrant--furious activity has not been translated into a genuine means of teaching people in ways that are intrinsically interesting--unless one counts as interesting CAL programs which are little more than programmed textbooks that happened to have found their ways onto computers. There is evidence that branching CAL does succeed when the objective is to teach either very simple, well-structured things or else provide drill and practice. However, when we attempt to do things that are more complex--such as teach writing skills that can be tranferred to essays--there is not much reason in the literature to be satisfied. Suppose, however, that we could build a "magic machine", that is, program a computer to take a sentence with errors from a student and return it to him perfectly corrected. What would be the instructional value of such a program?

At the very least, it would solve a number of quite practical problems inherent in typical CAI lessons in which a number of highly restricted responses are predicted and the student simply routed to the appropriate frame. One very practical objection to these lessons, for instance, is that they are really branching, programmed texts that somehow or other are being delivered by a computer instead of a book. There is little or at best conflicting evidence (Keller, 1979) that such programs really warrant the efforts and costs involved. Second, the canned nature of the lessons results in a rather troublesome inflexibility; if the student doesn't choose one of alternatives, the program can't do much for him. And any attempts to provide for more flexibility must quickly face the persistent question of just how much can be programmed, given the constraints of the programmer's cleverness in anticipating alternate answers and the problems of storage and retrieval. For as the computer lesson increases in its flexibility, in its ability to respond to a great many answers, one faces an inevitable combinatorial explosion of the things that must be programmed.

A more potent objection to branching CAL-- even from the vantage point of the most simplistic learning theory--is the notion that learning involves only discriminating among alternatives. For this must be the case when CAL limits the number of things it will accept as correct. Whether the student is asked to select from among explicitly offered options or whether he constructs a response that must come from a discrete subset of all possible answers- is irrelevant from an author's point of view; in both instances, the program is prepared to deal with only a limited range of answers. No doubt, a constructed response is preferable insofar as it forces the student to examine his repertoire without prompting. But should he generate an unanticipated answer, the logic of the constructed response mode is defeated (unless one considers it enough to say "your answer was not expected--try again"). Indeed, consider the following kind of dialogue:

Teacher: Johnny, how much is 2+2: 4? 5? 6? 7? 8?

Johnny: 3.

Teacher: "3" is not on my list of wrong answers. You have to choose only from the wrong answers I give you.

Johnny: But I don't like your wrong answer. I like mine. "3".

Teacher: Johnny, you're a poor student, and I'm going to put that on your report card.

Ludicrous as this dialogue may be, it is not entirely parodic since in fact branching CAL does refuse to speak to the student except on its own terms.

This refusal--or to be more precise, this inability--leads one to some fundamental theoretical objections. Again admitting that frame-based CAL has a place in achieving some very restricted teaching objectives in drill-and-practice, I suggest it cannot succeed in more complex instruction because it violates some basic cybernetic principles. The first of these--which is indeed a central cybernetic principle--is the Law of Requisite Variety (Ashby, 1956) which tells us that in order to cope with the variety of a system, one must have at least as much variety available as that system. A student must be seen in a system context as a complex configuration of entities who is made up of, interacts with, and is part of, other systems. As such, he generates a large variety of errors. In a teaching context, of course, one wants to attenuate this variety and reduce the possible number of responses to the correct one.* But frame-based CAL attenuates student variety right down to some variant of "A", "B", or "C".

Beer makes this point in his discussions about computers in modern society. They are generally used, he argues, precisely on the wrong side of the variety equation. That is, they don't respond to the truly important variety, the student and all the answers he might make but rather "accept much attenuated variety with a time lag" (Beer, 1974, p.40). Although he is not talking about CAL specifically, his words are directly applicable:

----

* Assuming we can always tell what is "correct": it is trivial perhaps in "How much is 2+2, not so in "What is the meaning of Hamlet?.

". . . It is the computer that generates the variety, and not the real world. This is quite fundamental nonsense. We are using our powerful tools to automate and to elaborate the limited processes that we managed to achieve with the unaided brain and quill pen--processes which our new tools were invented precisely to transcend."

Applied to the work surveyed here, Beer's comments are entirely to the point. In all cases, the programmer furnishes sentences, prompts, explanations, and so forth, while the student responds with a letter or a few words or positions a cursor. The control variety emanates from the machine, and as the author attempts to write more elaborate programs, the computer's variety is amplified and amplified again until it reaches the barrier of the combinatorial explosion. The student, meanwhile, sits rather quietly by. But it is his variety of needs and learning skills in which we are truly interested, and frame-based CAL does not react to that sufficiently.

The second great cybernetic principle frame-based CAL violates is a corrollary of Ashby's law: An optimal regulator works when it itself contains a model of the thing being regulated. This means that in order to teach someone anything, one must have a pretty strong idea of what that someone knows already, be it fact, fancy, or procedure. Again, Beer:

"A good pupil is trying to imitate his teacher . . . he is trying to build a model of what his teacher would do in given circumstances. A good teacher is trying to imitate his pupil . . .to build a model of his pupil's process, in order to understand his difficulties (Beer, 1979, p. 235).

Later on, this thesis talks about programs that do try to build models of students, but these, as we shall see, are very different than the sorts of programs described, and none of them teaches writing skills. In frame-based CAL, the typical student model is the one based on easy-to-gather information: right and wrong answers, the number of tries, time spent at a terminal (often down to milliseconds!), aggregate scores, and so on. What is not collected is exactly, however, what one would like to know: why did the student respond this way, how is his understanding structured, in what way has his variety been attenuated, how do the questions we ask him bear on what both he and we want to achieve. No one would minimize the difficulties in answering these questions; we labour under all sorts of constraints trying to understand our students. But as Beer puts it, "The constraint that maybe we don't know in the least what we are talking about is the most practical of them all--and the most likely" (Beer, 1981). Frame-based CAL does not know the students it teaches except in tne most superficial of ways-- criterion test scores, number of right answers, paths through a series of branches. It confuses what is easy to know with what is important to know, and so cannot build a useful model of the student it tries to teach. Without that model, it cannot regulate behavior.

The third major cybernetic failing of frame-based CAL is its lack of a metalanguage, a means of describing what happens through the whole of the learning process rather than just at its lower levels. Gordon Pask (1967), for example, recognizes at least three levels of discourse in computer tutorials with three corresponding languages: L0, L1, L2. L0 is the language most commonly found in CAL, and through it, the student responds to the questions of the tutorial program. L1, however, is a metalanguage that is used for discussing the strategies that exist at the level of L0. "L1 problem solvers reorganize, reconstruct, and repair the L0 problem solvers . . .". In turn, L2 discusses strategies that exist at the L1 level. This permits one to speak, in effect, of strategies about strategies. Pask's work is intimately connected with his epistemological framework of entailment structures, but even without incorporating them into a CAL lesson, one can see that there have to be ways of communicating between parts of the whole learning system in language capable of stepping outside, as it were, the immediate concerns. L2 can do this because it functions, says Pask, as an "executive" which "directs the attention of the student and allocates the 'resources' of L0 and L1 operations"

(Pask, 1969).

Interestingly, the TICCIT program acknowledges the need for such metalanguages. Bunderson and Faust (1976) note that "It is important to recognize that all major CAI systems developed over the past twelve years, with the exception of TICCIT, have worked primarily at Level 0. The drill and tutorial programs written in existing CAI languages have only occasional instances of Level 1 and Level 2 functions" (Bunderson and Faust, 1976, pp. 75-76). TICCIT's great emphasis on learner-control over the sequence of instruction comes from its objective of helping students become independent and develop better strategies. To that end, it has extensive "advising" built in to the system and uses a "metalanguage" to foster gains in learning strategies, attitudes, and responsibility. It does so aware that the sacrifice of author control may result in less effective learning. And indeed, evaluation of TICCIT's English course has shown that complex cognitive skills are not fully learned. My own suggestion is that it is not the cybernetic principle implicit in learner-control that is at fault but rather TICCIT's failure to observe the first two precepts for adaptive control, requisite variety and modeling. In any event, most CAL does not even begin to consider metastratgeies and metalanguages.

From the foregoing discussion, one may conclude the following:

1) In an effective CAL lesson, it must be the system that is teaching which responds to the student's variety, not vice versa. Therefore, a lesson must be able to accept any and all student input. This means that it must represent the subject matter in such a way as to provide a framework for interpreting anything a student might input;

2) In order to regulate the variety of the student and to be able to make the full range of requisite responses, an effective CAL lesson must have an accurate model of the student, one which reflects the pertinent substance and structure of knowledge rather than such things as scores on post tests.

3) An effective CAL lesson must do more than simply permit low-level exchanges about solving problems; it must enable the student to develop tactics, strategies, and even meta-strategies. And it must have some means of "conversing" at the appropriate levels of discourse among all its components (Boyd, 1971).

I believe these three considerations are essential in teaching complex cognitive skills via a computer. As I have said, perhaps the reason why low ability students benefit more than abler ones is that the skills they lack are ones simpler to describe and encode, so more amenable to programs that stress rote memorization or provide repetitive drill. As such, it is not so crucial to program them with vast amounts of sophistication. But without this sophistication, the higher and truly interesting cognitive skills are not going to be taught.

Satisfying the three cybernetic principles that have been described would produce the key elements in a general, intelligent program that would be free of the necessary rigidity of the frame structure. The student in such a program could generate sentences which then could be dealt with in ways far more powerful than frames. Something like this is done, in fact, with a number of programs which teach using some of the techniques of artificial intelligence (AI). (The following, for example: O'Shea (1979); Smallwood (1962); Kimball (1973); Tait et al. (1973); Goldberg (1973); Goldberg and Suppes (1972). But--to say the obvious--this is not so easily done with language instruction.

Now, in fact, we cannot use AI to build a "magic machine" for writing skills, and there is serious doubt that we ever can because we cannot fully manipulate natural language. There exists no satisfactory way of representing, for instance, all the contexts which humans use when they "understand" a sentence.* Indeed, the successes of AI have been confined to problems that have been carefully chosen and in which the key decisions (for example, what is or is not essential in a particular context) have been taken by people, not computers. Winograd (1972) is an excellent example of limiting to "microworlds" the context with which a program must deal. The solutions of Winograd and other AI workers are often brilliant, but they are also decidedly _ad hoc_ and so of limited generalizability. Whether or not this represents a dead end for AI itself is very much a matter of great dispute. It does mean, however, that transporting AI techniques to CAL is not done easily or (maybe) not at all.

But does this mean that one is forever confined to the CAL frame and all its dubious assumptions? I think not. For just as AI has developed _ad hoc_ solutions to its

----

* Hubert Dreyfus (1972) makes the case for the philosophic necessity of these contexts in his _What Computers Can't Do_.

problems, so too can I as a courseware author. My solution will not satisfy the AI worker, but that is because we are working on different problems. Neither will it satisfy the educational psychologist because I am less concerned with producing a theory of instruction than producing systems that teach people, even if one can't always say why. As an educational technologist, my test is a pragmatic one: does a system improve learning? This is quite different than asking whether the strategies of a CAL program are generalizable to notions about intelligence, machine or human.

OVERVIEW OF THE ATTEMPTED SOLUTION

The work described in the rest of this thesis is a proposed solution to the shortcomings of CAL that I have outlined. I propose to test the feasibility of constructing a system to teach English that can take any sentence as input and enter into a conversation with the student about its correctness. In order to do this, it will be necessary to model a properly-punctuated sentence and to develop programs that can use such a model to determine where the student sentence fails. The program must be able to see an analogy between what the student has written and what the model shows. The work also must develop a model of the student, both for each individual sentence he enters and for the series he enters over time. Finally, it will be necessary to develop conversational domains which can be used to discuss each input sentence and overall learning strategies. The result will not be a magic machine that will produce perfect versions of incorrect sentences.* It will, I think, contain the basic

----

*Indeed, if such a magic machine were available, it would have troubles of its own to resolve. Consider a student who knows very little about commas entering a sentence and getting back a perfectly punctuated version. What has he learned beyond the right answer? Is that sufficient? Should he have received partial corrections and some instruction? A hint? What? Human teachers often face elements to satisfy the criteria for an effective CAL system.

AN OVERVIEW OF THE REST OF THE THESIS

The work described in the next four chapters falls into three main parts: building an internal representation of the input sentence, modeling the student, and designing the exchanges between student and system. Chapter III discusses the various ways one might represent the knowledge needed to use commas properly. Chapter IV

---

these same questions which a comprehensive theory of instruction would answer. But we have no such theory.

describes SENTCHECK, the part of COMCOM that takes the student's sentence and creates the internal representation. Chapter V reviews some attempts to model the student in existing Intelligent CAL and presents the matrix model used by COMCON to measure the differences between the student and COMCON's own "master performer". Chapter VI describes the sorts of conversations open to a system that has gathered information of this sort about sentence and student. I also discuss some practical considerations of implementation and COMCON's "help" facility.

ORIENTATION

One point should be emphasized, I think. This is very much an exercise in educational technology, not in artificial intelligence, cognitive psychology, or natural language programming. As such, its chief concern lies in the pedagogical engineering problems one faces in putting learning materials on a computer. This is not to say that there is no connection between educational technology and any of those other areas; indeed, COMCON has many debts to all of them. But the kind of "quasi-intelligent" CAL described here belongs in territory rather different from artificial intelligence or cognitive psychology. The thesis does not attempt to build general theories of learning or intelligence as much as it tries to find engineering solutions to specific problems. It does take whatever other disciplines provide--transfer of learning, production systems, parts of Conversation Theory, and so forth--and tries to incorporate them into a working, viable system. Clearly, then, whatever is original here will not be of great moment to persons building theoretical models or language or learning. I hope, however, that courseware authors will find things of value, and it is to their concerns that I address myself.

CHAPTER III:

USE OF THE PRODUCTION SYSTEM TO REPRESENT KNOWLEDGE ABOUT COMMAS IN ADAPTIVE COMPUTER PROGRAMS

The formalism needed to represent a body of knowledge is not necessarily the best in formal terms but rather the most practically appropriate. Choosing one representation over another is very much a process of thinking first of the kind of knowledge involved and what one wishes to do with it; there is no philospher's stone of represention that will magically convey all knowledge in the best possible way. Therefore, the following discussion begins with a description of some quite specific tasks related to teaching the use of the comma, argues that the production system formalism is the most appropriate, looks at some possible alternatives, and also discusses a number of intelligent CAL programs that have used various representations including production systems.

My goal is to demonstrate the feasibility of a program to teach the use of the comma to college-level students. The starting point is a structural model (described in full in the Appendix I) of a properly punctuated English sentence. The model identifies the components of the sentence and shows how the comma combines with other items to mark logical relationships. Using the model as a guide, a student--or a computer--inserts or deletes commas under a number of specified conditions. For instance, one adds a comma between two main clauses joined by a coordinating conjunction or deletes a comma between a main clause and a following dependent one. So although the system deals with natural language, few of the usual AI concerns apply: It doesn't try to parse sentences into noun or verb groups; it doesn't try to generate a set of sentences from a finite number of words and a generative grammar; and it doesn't try to understand stories or even pronominal references. Rather, COMCON looks for certain features in any sentence a student might write and uses them as cues for correcting possible errors. Moreover, it assumes that the presence of a student supplies a sophisticated semantic processor (at least relative to any current computer ones) which COMCON can call upon from time to clarify the "meaning" of certain words or groups of words, just as the student can call upon the system for other types of clarification.

One is very much interested in procedural knowledge here rather than declarative--that is, the knowledge "how" we might do something instead of simply asserting the knowledge "that" something is true. We want the punctuator--human or machine--to behave in specific ways under specific conditions. This, to be sure, involves a certain amount of "knowing that" (and we must have a data base containing various facts or assertions); but eventually there has to be a procedure, a sequence of actions which will cause insertions and deletions to be made in the student's sentence. Therefore, whatever formalism we use must be capable of acting on data it receives--in this case, unanticipated sentences that students enter.

The production system formalism answers these requirements, but before looking at it, one might like to quickly survey a few other representations that have been used in "Intelligent" CAL programs. First, however, one might briefly get an overview of the representation problem by putting the following sentence in a learning context:* It takes Polly twice as long to get from Des Ecores to Queen Mary by Metro as by car, but if she travels at rush hour, there is no difference. If we were interested in "teaching" this in an arithmetic class, our representation would be a straightforward numerical model. But if our interest were in geography, we might want to show the streets by means of a map or other kind of graph. If we were interested, however, in the sociology of urban

---

* One must distinguish between the representation given to a student and the one a program uses to carry out some task. They may need to be formally related but they also can be quite different. In the representation focused on in this chapter, the learner's and the computer's are rather similar, both making use of the production system's "IF-THEN" structure.

environments, mathematical models and maps would not help us very much, and we might want some kind of "semantic net" (discussed below) to show the relationship between "car", "Metro" and "rush "hour". And if we were simply interested in determining if the sentence were punctuated correctly, we would need a model of a properly punctuated sentence that would have nothing to do with equations, maps, or semantic nets. The point, quite obviously, is that different sorts of knowledge require different sorts of representations in a teaching machine.

Winograd (1974) has suggested a number of criteria one must recognize in choosing a representational system. First, one must know how the representation will be used and how efficient it will be in operation. Efficiency can change drastically if the amount of knowledge increases. To look at all possible combinations in a three-element system is easy; thirty elements is not simply ten times more work, however, because the number of combinations increases exponentially. The second issue Winograd raises is learning: how does new knowledge get added? As system analysts know very well, an addition to one part of a system can have profound effects on other parts--often disastrous ones, at that. Purely modular systems (in which each element is independent) permit new items to be added simply through accumulation. However, such is not the case in most systems, which usually require pointers or indices. And one must always be sensitive to the potentially unpredicatble interactions when new things are added to a system. Winograd also notes that related to learning and adding things is the "naturalness" of the the representation; unless it seems natural to people, they will not like working with it if they work with it at all.* Finally, he speaks of the

* This is more properly a concern related to how people will use the system. A computer's internal representation can be mapped onto different sorts of interfaces. But because COMCON is intended to be used not only by students but by other authors who will extend it, the issue is worth noting.

problem of the very building of the system. He distinguishes between a "complex structure of many parts" and a "structure operating in a simple uniform way." One must tradeoff between them to reach an acceptable mix of complexity and generality, that ability of the system to deal with many different kinds of knowledge without collapsing, as it were, under its own weight.

Winograd also provides a list of operations a representation must be able to support. They are: _control_ (what should be done at any particular moment); _retrieval_ (what data might be used); _matching_ (does a particular thing apply to the task at hand and how so); and _application_ (what conclusions can be reached). In choosing the mix of features for a representation, one again trades off: explicit search commands can have much flexilibity (which increases the effective range of generality) or very little (which makes the system efficient). If one, for example, always knew what to expect, one could build in a high degree of control. But the real world is often full of unexpected events. Indeed, one could say that is the case for any interesting phenomena. A high degree of control might sacrifice the crucial capacity to respond to new things. Retrieval, that is, seeking out relevant data, presents few problems to highly structured programs since what they must look for is always specified. But, in the real world, sometimes the best one can do is call upon a set of heuristics which are unlikely to be especially quick, correct, or even resultative. The same sort of design issues are also involved in deciding how a system will seek a match and what it will then do with it. In brief, one can either limit a system to very simple problems (which undermines its usefulness) or try complex ones (which may lead to inefficiency or failure). Tradeoffs obviously are in order.

To illustrate, here is one frequently-encountered, representation in natural language and knowledge-based systems. One finds the semantic network (sometimes referred to as "semantic nets" or "conceptual dependency networks") in a variety of applications from story "understanding" (Schank, 1972) to CAL (Carbonell (1970)). A semantic net has been defined as "a system of interlinked concepts intended to capture meaning of a given expression or psychological phenomenon" (Boden, 1977). The usual rendering of a semantic net is a graph in which each node represents a concept and the arcs connecting the nodes represent relationships between the nodes. A typical relation might be "IS A": for instance, the word "and" is a "coordinating conjunction". But other relationships are possible although the representation, as Winograd points out, is not well-suited to complex formulae.

One can enquire about relationships among entities in the net; however, if there were no immediate connection between the nodes to answer these questions, one would have to search the paths between them until the points of intersection--if any existed-- were located. This would be done, as Winograd suggests, as if "by sending out signals from two nodes . . .spreading through the net one link at a time" until they met. But again, many nodes involve many searches, and a combinatorial explosion is always threatening. Of course, one need not search every node every time, but ways of developing heuristics to limit the search are not always going to be obvious.

Further, a semantic net does not always handle subtleties well. Boden shows, for example, that the typical IS A link is itself not strong enough to distinguish between "Mary is a girl" and "Fido is a spaniel" without additional information about spaniels being a subclass of "dog". Nor have nets been able to solve the problem of logical quantifiers like "all", "some", "every", and "any". Current implementations, Boden points out, have not begun to deal with these quite common ideas of everyday experience, despite the predictions that the conceptual dependency networks will lead to a breakthrough in understanding natural language. One can, indeed, develop schemes by which weights are assigned on the basis of frequency or other criteria. But deciding on these values is seldom likely to be straightforward.

These difficulties notwithstanding, semantic networks have found their way into CAL programs. For example, SCHOLAR (Carbonell, 1970) teaches geographical information such as the industries, climates, and populations of various Latin American countries. The data are represented as nodes on a semantic net so that a given chunk of information is viewed as a set of linked nodes. For instance, if a student wanted to know something about Peru (and in Carbonell's "mixed-initiative system" he can ask questions of the system as well as answer), the program searches for all the nodes connected to the one with the country's name. The following answer, for example, is given about Peru:

It is in South America;

the population is approximately 17,000,000;
the capital is Lima.

Provided that the information is explicitly encoded in the net, a great number of facts can be retrieved. But as Howe (1978) points out, implicitly encoded information --"What is produced in Lima?"-- requires additional rules such as "Find out what is the industry of a place in order to find out what a place produces." With sizable data bases, many such rules would have to be added. Further, using the system, as Gable and Page (1980) suggest, requires some skill of the student, and a student whose motivation was lacking might not request all that the system designers deemed relevant. A "Help" facility (or other kind of prompt) would be a useful addition.

Another semantic-net-based CAL program is one by Wexler which also teaches geography, this time about Canadian provinces. (And not always accurately: the Yukon is identified as a province.) The data are grouped into classes and then retrieved according to rules or "skeletal patterns". If the student's answer matches that produced by the program's search procedure, it is considered correct.

Both the programs of Carbonell and Wexler succeed reasonably well in presenting factual information to the student. But another program that uses semantic networks demonstrates the limits of the formalism. Brown, Burton, and Zydel (1973) produced a mixed-initiative program in meteorology which includes not merely facts ("What is fog?" or "What is precipitation?") but also deals with questions of causality (such as "What happens when the water temperature drops to 22 degrees?"). The former sort are roughly comparable to the geographical information seen in Carbonell and Wexler, but the latter require a process to be represented and some computation to be done. Moreover, the program is sensitive to context, and various values of phenomenom can change in the course of a session. Hence, what is required is both a dynamic and a static model of meteorology.

The static model used is a semantic conceptual dependency network with three basic kinds of nodes: objects, concepts, and processes (although this last is restricted to factual representations such as their definitions, not their actual to functioning). For the dynamic model, the authors represent processes and their interactions with each other as "augmented finite state automata". Each automaton has a specific calculation to perform and the cross product of all automata "forms a global automaton which functions as the dynamic model of our meteorology data base" (p.250). New automata can be added as the model increases in complexity. In effect, then, the program has two different kinds of knowledge representations--one to produce factual data and the other to calculate outcomes.

The significance of this work is that it combines divergent models since more than one order of task must be carried out. The semantic network's strength is in the way it can relate facts: its weakness is in its problems handling dynamic relations easily. This does not eliminate it from consideration, but points out that it should be combined with other appropriate representations depending on what one wishes to do.

In fact, production systems have been combined with semantic nets, and Waterman and Hayes-Roth (1978) cite about a half dozen cases. Hayes-Roth (1978) points out that production rules (considered below) can be "represented as nodes in a network containing specifications about what actions to take if a message is received from input arcs to the node". Again this emphasizes that at some point, actions have to be taken. COMCON does in fact use a series of small semantic nets because there are a number of general to specific relationships which the structural model of the properly punctuated sentence must know. For instance, a typical search is for a coordinating conjunction; this is done by looking for all of a group of words that fall into this class ("and", "but", "or", etc.). The same is true for classes such as subordinating conjunctions, prepositions, determiners, and so forth. A production system with an embedded semantic network makes it possible, therefore, to group related words but also to perform operations like inserting or deleting commas.

As Howe (1978) points out, most examples of machine intelligence are exhibited in CAL programs for mathematics and other well-specified areas. This makes the representation problem somewhat more tractable. Goldberg (1973) uses an AI technique of theorem-proving for lessons in both algebra and first order predicate calculus. Goldberg and Suppes (1972) use "resolution-based theorem proving" to help student construct and verify logical proofs. And Kimball (1973) has written a program for integration problems in algebra. But the mathematical and logical domains involved in these three programs are quite different from natural language, and the literature of CAL does not show work in language along these lines.

In other, less mathematically-based CAL, there are a number of simulation programs such as those in criminal justice and land use, although there have been problems in designing ways for input to be easily entered by users (Gable and Page, 1980). One of the most successful simulation programs is SOPHIE (Brown and Burton, 1974), an electronic troubleshooter that presents the student with a circuit schematic that has some particular fault and helps student isolate and correct it. The difference in the success between SOPHIE and the other simulations may be because SOPHIE works in a domain--the structure of electronic circuits--which is more clearly specified and certainly better understood than land use or legal issues. When the threshhold between demonstrably formalizable and as yet unformalized experience is crossed, modeling and the CAL task clearly becomes much more difficult.

Two other representations might also be briefly mentioned. Predicate calculus has been used extensively in, mathematics and formal logic and is a powerful formalism for representing "knowledge" in very general ways. Typically, some problem is approached by taking a group of axioms and looking for a proof in the problem space. However, this can often take a great deal of time unless there is a rather small set of facts to be manipulated. Winograd (1974) places the typical number at less than a hundred and often less than ten. Given the large number of possible sentences a student might enter, using predicate calculus would seem likely to be an inefficient approach to teaching punctuation. Moreover, predicate calculus is neither particularly simple to use nor particularly congenial.

Another possible representation is a procedural programming language like PLANNER which attempts to be more flexible than typical computer programs while yet retaining their efficiency. PLANNER (and similar languages) contain a set of assertions and a set of theorems which are combined as procedures rather than declarative statements. The section on production systems below, shows this same emphasis on procedure. Both PLANNER and production sytems are pattern-directed; that is, they look for patterns in the input to control their actions. But PLANNER checks only particular assertions, not the whole input, which suggests that looking at the whole of the student's sentence is much simpler with a production system than with PLANNER. And this is not to discount the impressive work in understanding language that has been done by such programs as Winograd's SHRDLU (1972). (Although, one should note that SHRDLU is not a CAL program but one that demonstrates natural language manipulation.)

The kind of formalism I think a sentence punctuator needs does exist under the general rubric "Pattern-Directed Inference System", more commonly referred to as a "production system". (Recent literature refers to the production system as a type of pattern-directed inference system, less powerful and more narrowly defined.) Such systems are first and foremost organized on the basis of data or events; that is, they respond to varying kinds of data presented to them. Such data need not be of a single or particular nature and the system need not respond in inflexible ways. Instead, patterns in the data cause different portions of the system to be activated. Waterman and Hayes-Roth (1978) have identified three basic components of such systems: 1) "pattern-directed modules" (PDMs) which are "fired" by patterns in the data; 2) data examined by the PDM and perhaps modified by it; and 3) an executive or interpreter which acts as a control in deciding which PDM should be used in case more than one is possible.

Typically, data examination and modification are separate tasks, and one can most conveniently represent them by "IF-THEN" formulations. That is, one usually encounters pairs of "antecedent-consequent" rules in which data are examined according to patterns in the antecedent (or left side of the IF-THEN statement) and modification is performed in the consequent (or right side). For instance, one might find the following:

IF the data contains an x,
THEN change the x to a y.

The program examines data by searching for the patterns of the antecedent portion of the production rule, and such patterns can take a variety of forms--strings, networks, graphs, lists, and so forth. The program can modify data by asserting that something is true (IF A is there, THEN B is true) or by causing various actions to take place (IF A is there, THEN do C, D, and E). Therefore, once a match between a production rule and some portion of the data has been found, either an action or an assertion can be executed. If several possible matches are found, the executive must invoke a strategy to choose which of them will be performed first*.

Newell and Simon (1972) give a number of reasons why the production system is a good model of human cognition. They stress its computational generality, the ease with which new information in the form of rules can be added to a system, and the way it reflects the organization of human memory into long and short term components (with the data being similar to the elements in short term memory and the production rules like elements of long term memory). And indeed there have been many different domains in which the production rule formalism has been

----

* Unless one can perform them simultaneously in parallel processing systems.

used successfully. One sees it in symbolic logic, Markov algorithms, Chomsky rewrite rules, and, of course, in many programming languages. Waterman and Hayes-Roth cite the following applications: program synthesis (that is, developing computer programs once having supplied an "exemplary program"); computer system architecture (special languages with pattern matching capabilities); knowledge acquisition (in which human experts convey their expertise to a task-oriented sytem); knowledge engineering (application of artifical intelligence research to a range of real world problems); natural language understanding; and cognitive modeling. Some of these applications are discussed more fully below, but for the moment, the point is that the production system is a formalism of sufficient power and generality to have been investigated by many different researchers in many different areas.

Some idea of this power and generality may be seen by noting that productions systems are useful not only to establish a hypothesis (IF A and B THEN C) but to confirm one (IF C THEN A and B). This latter use is called "backward chaining" (Waterman and Hayes-Roth, 1978) because one moves from the consequent (that is, the right hand part of an IF-THEN statement) to the antecedent (or left-hand side). Thus one can work either from a set of conditions (IF there is a coordinating conjunction, IF it joins two main clauses, THEN add a comma) or from a conclusion back to a set of conditions (IF a comma is correct, THEN there must be a coordinating conjunction and two main clauses). Data, then, can be approached from opposite (but complementary) perspectives.

One can elaborate production systems in a number of ways. The most obvious is that the number of rules can be increased over time (as when a series of experts add new information about some problem), and one does this without much difficulty. This is because production systems are modular: data and rules are quite separately maintained and adding a new rule does not affect the validity of another. Secondly, the control structures are also modular; one can add a great many kinds of "metarules", "filters", or "gatekeepers" which decide what actions the system should take and what data it should look at first when a number of possibilities exist. This can significantly increase program efficiency.

Without going fully into the details of COMCON, let me show why the production system represention is in fact appropriate when using a computer to teach the comma. Firstly, any teaching system must be able to respond to a wide variety of possible student inputs. The chief failure of the typical branching CAL lesson is its inability to process any but a small number of possible responses. The production system, on the other hand, looks at anything a student may enter. This is especially important in teaching punctuation because we are not primarily concerned with the meanings of words (and so do not need a vast dictionary in which to look them up) but in the relationships among parts of a sentence. These are frequently marked by a set of about a hundred function words like conjunctions, determiners, prepositions, and so forth. One can easily accommodate these words as parts of production rules and match the data (that is, the student sentence) against their indexing patterns. Moreover, the modularity of the production system permits two important features: 1) complex conditions can be called (for example, IF there is an "and" OR "but" OR "or" OR "for" AND IF there is also a main clause OR two equal elements OR a series AND if there is no comma . . .); this, in fact, reflects the complexity and variety of English sentences. And 2), as new rules are discovered, they can be added, giving the system designer the chance to modify and enhance the system at any time in the future. Indeed, my experience in building COMCON has been to add rules one-by-one, over long periods of time as either better understanding of the material evolved or as I extended instruction to reach new objectives.* One begins by doing an attribute analysis of a sentence (it has a subject, a predicate, it has two clauses, and so on); but these

----
* The present system now begins by looking for coordinating conjunctions because classroom observation showed that many comma errors begin there. The original version started with a scan for extraneous commas and did not consider prepositions, for example.

analyses are not going to be complete or even correct the first time through. As new ideas about sentences are formed, they must be integrated into the stock of old ones. The production system is ideal in this respect since its modularity allows new rules to be added without, as we have seen, affecting those already there.

Production systems are also very useful because they handle powerful, logical manipulations in fairly straightfoward language (the "IF-THEN" statements) without complicated notation. (Compare, for example, having to use predicate calculus.) One often needs to understand causal relationships when punctuating: for example, the presence of a preposition at the beginning of a sentence "causes" the delay of the main subject. That is, except in very special locutions, a prepositional phrase cannot contain the sentence's subject. So if one were to find a preposition in the first position of a main clause ("From all I read, things are really bad"), one would know that the subject will be delayed and a comma must be used to alert the reader. (This will be shown in full detail when the model itself is discussed.)

This is a straightforward operation for a production system which simply must look at the first group of nonblank letters, determine if this string matches any of the antecedent sides of its production rules that concern prepositions, and then look ahead to the rest of the string to see if the writer has or has not included a comma.

A second kind of reasoning ability easily accommodated by the production system is logical implication. Again, here is a fairly trivial example: COMCON frequently needs to assign Boolean truth values to variables which may later have to be accessed either singly or in groups. In the example above dealing with prepositions, the first group of nonblank letters is examined to see if it is a preposition. The results of this search are assigned to a variable in the form of a 1 (if the search was successful) and 0 (if it failed). In terms of the production system formalism, one can say: IF A (that is, a preposition exists) THEN B (a variable is assigned 1). There are doubtless many other ways of achieving the same thing, but this way is surely among the simplest and most congenial.

Congeniality brings up an important point--the interaction between system designers and programs. Because the production rule formalism reduces complex problems into modular and relatively short sub-problems, it is not difficult for human monitors to understand what is going on in the system, how to debug it, and how to add to it. Documentation, for instance, can be attached to each production rule, making maintenance very easy for someone other than the original designer.

At this point, it might be helpful to describe some computer programs which use the production system formalism. The MYCIN system (Davis et al., 1977) helps physicians decide which medication to prescribe for various kinds of bacterial infections by focusing on the particular disease, thus obviating the need for a "shotgun" treatment with broad spectrum antibiotics. MYCIN is not intended to replace the physician but to help him towards diagnosis, and even though it does not technically qualify as a CAL program, one can readily perceive it as a tutor.

MYCIN starts by gathering basic data about the patient--sex, age, preliminary reports of tests, and so on. It then begins to focus on specifics--what is the shape of the organism being traced, how did it grow, where did it likely enter the body. The questions become progressively more narrow as MYCIN moves towards its decision because it forms a number of reasonable hypotheses based on preliminary questions and then seeks confirmation. This is done through the approximately 300 production rules the system contains of the following kind:

IF the infection is primary bacteremia

AND IF the suspected entry point is the gastrointestinal tract

AND IF the site of the culture is one of the sterile sites

THEN there is evidence that the organism is bacteroides.

Note that the consequent part of the rule is entertained as an hypothesis, and questions asked by MYCIN try to establish the existence of the antecedents. This is the "backward reasoning" I spoke of earlier. It would be possible to establish the presence of all the antecedents first (that is, all the discrete facts about the symptoms), but the backward reasoning permits a more focused questioning by the user--one possibility at a time--which again is quite congenial and natural for human beings.

Moreover, MYCIN does not present itself as an oracle on high. It asks the doctor to enter values at various points indicating how certain he is of the salient factors of the situation. This helps to establish some minimal levels of certainty before reaching final decisions. Further, MYCIN is prepared to converse with the doctor (in a subset of English) about why judgements were made at various places. This, in turn, permits the doctor to override MYCIN, and indeed permits the system's designers to add new production rules should it become clear MYCIN is in error or incomplete. The modularity of the formalism makes this a relatively straightforward task.

To give one the sense of the exchange between MYCIN and the user, here is a brief sample of the program in operation:

WHAT TYPE OF INFECTION IS IT?

Primary Bacteria.

WHEN DID THE SYMPTONS FIRST APPEAR?

May 5, 1975.

FROM WHAT SITE WAS THE MOST RECENT POSITIVE CULTURE TAKEN?

From the blood.

CAN YOU IDENTIFY THE FIRST SIGNIFICANT ORGANISM FROM THIS BLOOD CULTURE?

No.

IS IT A ROD OR A COCCUS OR SOMETHING ELSE?

Rod.

WHAT IS THE GRAM STAIN OF THIS ORGANISM?

Gramnegative.

The exchange continues until MYCIN is ready to make a recommendation:

DOES THE PATIENT HAVE ALLERGIES TO ANY ANTIMICROBIAL AGENT?

No.

WEIGHT OF PATIENT?

70 kilograms.

MY FIRST RECOMMENDATION IS AS FOLLOWS:

GIVE GENTAMYCIN USING A DOSE OF 119 MG(1.7 MG/KG) Q8H IV FOR TEN DAYS. MODIFY DOSE FOR RENAL FAILURE . . .

This sample is very much abbreviated, but one can still see how there is a cooperative effort between human and machine. Each provides what the other cannot, or at least, not conveniently.

In brief, MYCIN offers an excellent example of the use of a formalism that is at once powerful and flexible to solve nontrivial problems in a manner most accessible to humans. It does not replace the doctor; as Boden (1977) points out, it augments him. This is quite similar to COMCON which does not intend to produce a corrected version of the student's sentence but to help him correct it himself. In both MYCIN and COMCON, the key element is cooperative interchange between two sorts of processors--human and machine--being brought to bear on the same problem.

Another successful problem solving program using production systems in DENDRAL (Buchanan, Sutherland, and Feigenbaum, 1969) which identifies unknown chemical compounds by interpreting the results of a mass spectrometer analysis. It carries on in much the same way a trained chemist would--that is, it makes a reasonable hypothesis after some preliminaries and then tests it against a set of conditions it "hopes" to find. In practice, these hypotheses are lists of possible structures for the substance which are checked against the mass spectrogram until a match is made. Here is one example of DENDRAL's production rules:

IF there is a high peak at atomic-number charge point 71,
AND at 43 AND at 86 and ANY peak at 58,
THEN there must be a N-PROPYL-KETONE-3 substructure.

There are a few hundred such production rules in the system, and one can clearly see a very sophisticated level of expertise in the knowledge base. But the way that knowledge base is represented--via the production rule formalism--allows for a clear and cogent expression. And indeed it performs its intended job with the competence, its developers note, of a good graduate student.

In summary then, the production system formalism has shown itself both powerful and convenient in a variety of applications. It provides the ability to mark the relationships between ideas that a semantic net does but with considerably greater ease in manipulating them. It is preferable to a procedure-based programming language since it looks not at single assertions but at the whole input. And it is far simpler to use than predicate calculus even if one could solve the problem of the latter's relative inefficiency when dealing with large data bases. The production system's modularity allows the addition of new rules over time and makes the program easy to maintain. Experience with successful projects like MYCIN and DENDRAL-- even though both are not strictly speaking CAL in the sense proposed here-- suggests that a production system should be powerful enough for teaching punctuation.

To this point, we have considered systems for representing knowledge only insofar as they were directly relevant to analyzing input sentences. However, another part of the system --the HELP facility described in the last chapter-- has a quite different goal. HELP is a series of some twenty self-contained modules which can be accessed by the student at any time and which provide lessons about various topics related to punctuation. Taken together, the HELP modules constitute a fairly comprehensive minicourse which could be used independently of COMCON. The way these modules might be organized suggests a very different problem, however, than scanning sentences for key features. Therefore, rather than production systems, one needs something quite different.

The notion of "entailment structures" has been described in several places by Pask and his associates. (See, for example, Pask (1975b) for a particularly detailed account.) An entailment structure is a map or graph of a particular subject matter domain which meets the requisite demands of coherence, consistency, and cyclicity. This means that it provides a student with a full image of a subject matter, with each topic in that domain clearly linked to the others from which it is derived. In practice, this last permits a student with partial "understanding" (a word Pask uses in a special way) to discover for himself the meaning of a given topic. Further, the entailment structure contains both theoretical topics of the subject matter as well as analogous, parallel topics from the world of everyday experience.

One begins to construct an entailment structure by having a subject matter expert descibe the essential topics in his field and the relationships that exist among them. From this, a relational network emerges as a graph whose nodes show the topic relations and whose labeled and directed arcs show paths between these relations. This evolves by a set of transformations which can be assisted by special course assembly routines into a final structure that has been "pruned" (that is, one in which topics are represented as hierarchies) and with which the student interacts. One should see that despite the final hierarchical arrangement, the graphs themselves do not enforce a particular path through the subject matter. Indeed, pruning takes place only after the student indicates his learning goals or aims. Until that time, the entailment structure simply is an image of a domain with the full range of relationships present. This is an important distinction, for in order to allow the student eventually to demonstrate his understanding, it must be possible to derive any single topic from another. With the domain map in place, the expert goes on to provide fuller descriptions of the relations between topics as well as devising a set of tasks that permit one to perform the procedures that translate "knowing" to "doing". The understanding of topic can be shown either by verbalizing or else working with a modeling facility.

The student, having been given an entailment structure, can work through it in the paths that he thinks appropriate. This could take the form of exploring various nodes or aiming towards understanding of a particular topic. If he has all the prerequisites for the topic at which he aims, it becomes his goal. The process is much like going through all prerequisite subtasks until the end task is achieved.

Pask and his associates have developed entailment structures for a number of different subject matters, ranging from probability theory, biology, chemistry, phsyics, even history (a discipline whose methodology and procedures are not generally as well enunciated as the others). Whether it is, in fact, possible to develop an entailment structure for the topics involved in punctuation is not clear at the moment because the necessary "cyclicity" (that is, the way each topic can be derived from others "bi-directionally") is not evident. Here, for example, is a list of the topics which constitute necessary information about a properly-punctuated sentence: introductory elements, main and dependent clauses, nonessential elements, series, and coordinating conjunctions. To these, HELP adds modules for phrases, subordinating conjunction, prepositions, clauses, dependent clauses, subject, predicate, noun, verb, and stock expressions. In some cases, one sees the necessary cyclicity easily. For instance, an introductory element can be either a dependent clause, a prepositional phrase, a single word or a stock expression. One can, therefore, think about any of them in terms of any other. The same is true for clauses: one could speak about main and dependent ones, subjects and predicates (which all clauses have) and subordinating and coordinating conjunctions. Again, there is a local cyclicity here. However, if one were to move laterally across an entailment structure of this domain, it is hard to see what is the necessary connection between a nonessential item and an introductory clause--except perhaps that they are both constituent parts of possible English sentences. Similarly, one might ask what is the necessary connection between a series and the subject of a sentence, unless the particular subject of a sentence itself happens to be a series of some sort.

Now these may well be non-objections because facilities such as THOUGHTSTICKER (Pask,1976) exist which might enable one to produce an entailment structure of punctuation topics to satisfy the demands of coherence, consistency, and cyclicity. However, such a facility is not available at the time of writing.[4] Future development of the HELP package could be in the form of giving students entailment structures of the material and letting them work their way through it. For the time being, in addition to having a menu of topics available, individual HELP modules refer to others and encourage students to call them as the need arises. Future releases could also include TICCIT-like "advisors" or Ausubel's (1968) "advance organizers". This is clearly an area open to more study.

CHAPTER IV

SENTCHECK: BUILDING AN INTERNAL REPRESENTATION OF SENTENCE PUNCTUATION

What must we know if we are to punctuate a sentence properly? What does a master performer (that is, someone who is an expert at punctuation) do when confronted with a sentence which he either must accept as correct or change? Such knowledge is not easy to discover because punctuation seems to most people--even good writers--something done rather automatically, indeed, even intuitively. Worse, experts are often inconsistent in their procedures (or at least apparently so) and will often disagree with each other. But any CAL program that wishes to teach punctuation must have this "knowledge", especially if it is going to do so in an intelligent way. So what follows is not an attempt to legislate conformity but to make explicit and coherent the kind of knowledge that an intelligent CAL punctuation program must have.

The first thing we must know is what a properly punctuated sentence looks like from the perspective of logical intelligibility. That is, we must have some kind of general model which shows the various constituents which may be present in a legitimate sentence and how they are related. For many writers, this is frequently ignored or thoroughly internalized; but in either case, it seldom if ever gets mentioned. Moreover, even with good writers, often there seems to be a mystery about just what punctuation does. Contrary to what most students wrongly think, in discursive prose a comma does not indicate a a speaking pause although it may coincide with one. Rather, the comma functions as a logical marker that helps show the reader the relationships among different parts of a sentence. Here is a model of a properly punctuated compound-complex sentence:

(I1,), MC1(DC1), cc (I2,b) MC2(DC2)(,NE,)(S)

where MC1 and MC2 are main clauses, DC1 and DC2 are subordinate clauses which depend on MC1 and MC2 respectively; I1 and I2 are introductory elements for MC1 and MC2; cc is a coordinating conjunction; NE is a nonessential element; and S is a series. Items in parentheses are optional and NE and S can be shifted to other positions within the sentence. Appendix I gives the full derivation of this model.

The model is essentially a structural one, but syntax by itself is insufficient since punctuation also requires semantic information. It is not enough to know that the various constituents relate to each other in some way; we also must know what are the essential attributes of those constituents. For instance, to know that a following dependent clause should not be separated from its main clause by a comma means little if we don't know what constitutes either of them in the first place. That kind of knowledge is of two sorts: partially semantic (that is, having to do with meaning of the whole utterance) and partially patterned (that is, having to do with recognizing the presence or absence of certain key features of English).

Here is an example of semantic knowledge and patterned knowledge coming together. We know that a dependent clause, for instance, cannot stand by itself. _When the noodles burned_ requires a main clause of some sort to complete its meaning, to change it from a sentence fragment. We "know" that--in part--semantically: the clause refers to some unspecified event (perhaps _Polly cried_) for which the reader immediately begins to look. But _When the noodles burned_ begins with a subordinate conjunction which also points to the fragmentary nature of the clause. There are about twenty such subordinate conjunctions (and a few relative pronouns that operate the same way), and all dependent clauses begin with them. Knowing this, it then becomes a simple matter to look for one of them to see whether a sentence contains a dependent clause. The trick for learners (or programs) is being aware of this; the trick for instructors (or programmers) lies in alerting students and machines. The output reproduced in this chapter suggests some of the ways one may do this. (They do not, however, constitute the conversations that will eventually take place and which are described in the last chapter.)

One can, in fact, do something like the same kind of search for a number of other things. Morever, having searched for these markers (or features or attributes), it is then possible to note their configuration in any sentence and so make inferences about the correctness of that sentence. For example, the model says that there should no comma between a main clause and a following dependent clause. Should a search find a subordinate conjunction somewhere in the middle of the sentence that is immediately preceded by a comma, we know an error likely has occured. (The appendix explains why it is "likely" rather than absolutely so.) Or should we find a coordinating conjunction between two main clauses that is not preceded by a comma, again we can make the inference that an error has been made.

Perhaps the most straightforward way of discussing the number and variety of these inferences is to describe the computer program that performs them. It is convenient to do so because a computer program not only must "know that" (for instance, that a dependent clause begins with a subordinate conjunction, that "when" is a subordinate conjunction, or that there is a comma before some word), it must also "know how" (how to determine if a string is a dependent clause or how to find out if a comma precedes it.) A discussion in terms of a program is also useful because the particular application of these ideas is, of course, a group of CAL lessons to teach punctuation.

SENTCHECK is a set of APL functions which takes as input any sentence a student might enter, forms an internal representation of that sentence which in turn is basis for COMCON's conversations with the student about the correctness of the sentence. The theoretical basis of the program is the production system formalism already described. The program takes the whole of the student's sentence as its data and performs operations on it according to the various indexing patterns of its production rules. These rules, given below, are a series of "IF-THEN" statements which have resulted from the attribute analysis described in Appendix I. For now, the focus is on those production rules which constitute the system's internal represention of the model of a properly-punctuated sentence. The responses given by the computer are not those which the student finally sees. (Those require an extensive discussion of, among other things, conversation theory and tutoring strategies.) SENTCHECK makes important use of a number of APL's best features, particularly its interactive nature, its handling of arrays, and its global and local variables. The thesis does not include code, but this chapter will instead talk about the APL functions in some detail and provide an overview of the logic of the program.

Figure 4.1 provides an overview of SENTCHECK which begins when function CONTROL is called.

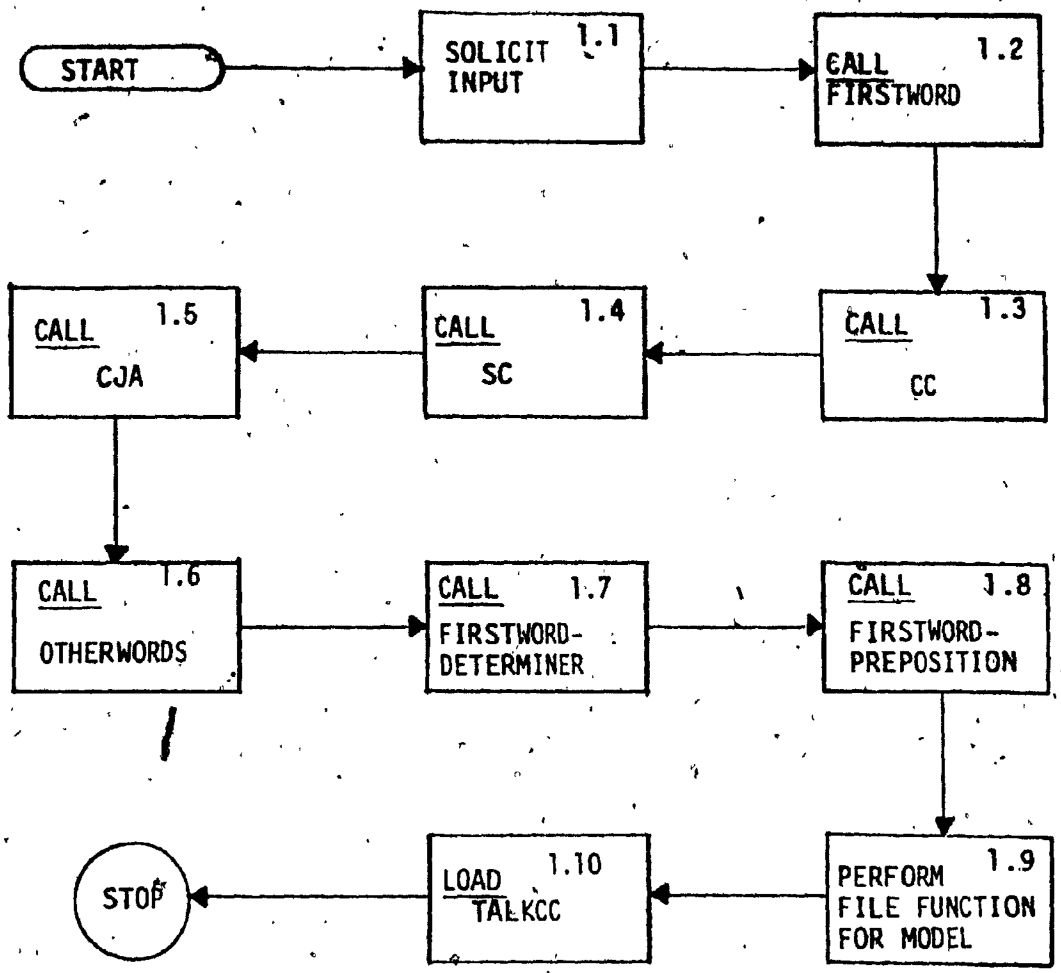


FIGURE 4.1: Control Function for SENTCHECK workspace

(In fact, CONTROL is a "latent function", one that starts as soon as its workspace is loaded. This makes the system extremely "user-friendly".) CONTROL, of course, excecutes its lines serially, but the structure it imparts to SENTCHECK is more hetrarchical than hierarchical. There is no necessarily fixed order for the functions it calls since the rules about commas are independent of each other. SENTCHECK begins with coordinating conjunctions because so many punctuation errors revolve around them and the structural elements they join. Indeed, some functions may be discarded altogether and some may be called more than once depending on what the input sentence looks like. Put another way, the input causes the functions to produce results of different sorts, and CONTROL acts as a clearinghouse to share the information generated. Rather than force a single decision on what SENTCHECK will do next, CONTROL keeps the full range of potential actions available.

CONTROL begins by asking the student to enter any sentence of his own choosing. The sentence is immediately assigned to two variables, ANS and SENT: ANS is a buffer whose value will change from time to time while SENT is permanent record. CONTROL then calls upon function CC which begins a mapping of all coordinating conjunctions the sentence might have (that is, all instances of "and", "but", "or", "nor", and "for").

CC does this by seeking all instances of the strings "and", "but", etcetera, checking to make sure that there is a blank before and after each so that it will ignore words that contain them (like "candy" or "butter"), and then sorting the results of its mapping to coincide with the order in which the words appear in the sentence. CC (and all the search functions discussed below) actually returns a "2 by n" matrix ("n" being the number of coordinating conjunctions in the sentence); the first row in each column shows the position of the first letter of each word, and the second row shows the position of the last letter of each word. This done, CC then checks whether a comma precedes or follows any item it has mapped and assigns either a 1 or a 0 (for true or false) to an appropriate variable. It can therefore map all coordinating conjunctions in a sentence (up to a maximum of five--a reasonable number) and see how each has been punctuated. This information is important because a coordinating conjunction followed by a comma is always wrong; a comma before, however, can indicate either the presence of a second main clause ("The noodles burned, and Polly cried") or an error ("I like noodles, and cheese"). All this information is stored temporarily and will become the basis of the conversation with the student.

CC also calls three other functions: NEXTWORD, WORDBEFORE, and WORDBEFOREBEFORE. As their names suggest, they search out the words which precede and follow any particular coordinating conjunction. These three words are especially useful to know if the coordinating conjunction has been used to join the last element in a series of single words. (A series of groups of words ("From the hill, from the dale, and from the valley") must be handled by asking the student for the words that begin and end each string.) Strictly speaking, this may not be necessary since CC also checks to see if there is a comma before a coordinating conjunction, and this information can be used to decide whether the punctuation of a series is correct.

CONTROL then hands the sentence to SC, a function which operates like CC but instead maps subordinating conjunctions. These include "when", "where", "if", "because", "although", and about ten others. (Some relative pronouns like "that" and "which" act like subordinating conjunctions and are therefore included in the mapping.) SC also searches for commas preceding subordinating conjunctions because they either indicate an error or that the writer wished to make a clause nonessential. Again, this cannot be determined without other information which will have to be collected in COMCON's exchanges with the student. SC also notes whether any of the subordinating conjunctions it has mapped begins the sentence. This is important since a subordinate clause which acts as an introductory element must be followed by a comma.

CONTROL then calls CJA to map conjunctive adverbs ("moreover", "nevertheless", "therefore", and so on). These words are important because very often they act as transitions in logic between parts of a sentence ("I think, therefore, I am") or begin main clauses ("However, dolphins also think"). A conjunctive adverb must be preceded by a semicolon when it separates two main clauses. CJA, therefore, searches for semicolons and commas associated with conjunctive adverbs.

Next CONTROL calls OTHERWORDS which looks for any words other than subordinating conjunctions, conjunctive adverbs, or coordinating conjunctions that are associated with commas. As the coordinating conjunctions, subordinating conjunctions, or conjunctive adverbs are mapped, a record is made whether there are commas associated with them, and the map of all the sentence's commas is reduced appropriately. This allows one to know where other commas are in the sentence, and so can permit COMCON to discuss them with the student.

Finally, CONTROL calls FIRSTWORD which simply notes the first word in the sentence. This information is then passed to FIRSTWORDDET and FIRSTWORDPREP to see if the that word is either a determiner (for example, "a","the", "some", and so on) or a preposition. A sentence that begins with a determiner must always have that first noun phrase as its main subject. This means there must be no comma following it since there is no introductory element. (Consider, for instance, "The man ate grapes" or "Some men are called Socrates"). Conversely, a sentence that begins with a preposition must have its subject following the introductory element which that preposition starts (as in "In winter, birds fly south"). The only exception to this is when a preposition is used as part of a noun phrase, as in "Of alligators was all my interest", a construction not often encountered in student writing. The word "to" at the start also presents some problems since it can be either a preposition ("To Toronto, the plane takes three hours") or a part of an infinitive ("To do good was my intention"). Once more, this is information that COMCON must get by asking the student. A last program, FIRSTWORDRELPR, looks to see if either "what" or "that" begins the sentence; since their presence would indicate the sentence begins with a noun clause ("What I need is hope"), this again means the sentence has no introductory element before its main subject.

Let us examine some of the output that emerges. The student's words are in uppercase letters and the computer's in lower case. Note that the final implementation does not give the computer's responses as they are here; their present form is simply to give one an idea of the internal representation of the sentence that was used in earlier versions of COMCON.

```
enter a sentence:


WHEN THE CHILDREN SAW THE CANDY THEY WERE HAPPY.

there are no coordinating conjunctions

the subordinating conjunction is "when"

it starts the sentence
```

there are no commas in the sentence

because the sentence begins with a dependent clause, it needs a comma before the subject

That there are no coordinating conjunctions means that the sentence is made up of only one clause, and that there is neither a series, a compound subject, verb, nor object. Because the subordinating conjunction starts the sentence and there are no commas, it is clear that the introductory dependent clause needs to be followed by a comma.

enter a sentence

THE CHILDREN WERE HAPPY, ALTHOUGH THEY SAW THE CANDY.

there are no coordinating conjunctions in the sentence the subordinating conjunction is "although"

(and it has a comma before it)

the comma before the dependent clause is wrong

The absence of any coordinating conjunction again shows a simple (as opposed to a complex) sentence). The subordinating conjunction, however, coming somewhere other than at the start of the sentence shows that the main clause begins the sentence. This is further confirmed by the presence of the determiner "the" at the start. And because there is a comma before the subordinating conjunction, SENTCHECK can infer that the sentence does not follow the model which says that there can be no comma between a main clause and a dependent one that follows it.

enter a sentence

FROM EAST TO WEST THE COUNTRY WAS AGREED IN ITS OPPOSITION WHERE THE CONSTITUTION WAS CONCERNED.

there are no coordinating conjunctions

the subordinating conjunction is "where"

there is no comma in front of it

it does come at the start of the sentence

the first word is "from"

it is a preposition

there are no commas in the sentence

insert a comma after the introductory element

Again, the absence of a coordinating conjunction but the presence of a subordinating conjunction shows the sentence is complex. Because there is no comma before the subordinating conjunction and since it does not come at the start, the punctuation between the main and dependent clauses is correct. The sentence does start with a preposition which means that the main subject does not come first. And because the sentence lacks commas, we know there should be one after the introductory prepositional phrase.

enter a sentence

I LIKE CANDY AND TEA, BUT SHE DOESN'T CARE ABOUT BEER, BUTTER OR BISCUITS.

here the list of coordinating conjunctions and

but

(with a comma before it)

or

there are no subordinating conjunctions

the first word is "I"

it is a pronoun in the subjective case

"I" is the subject of the sentence

Here we must deal with three coordinating conjunctions, and they are assigned variables cc1, cc2, and cc3. Moreover, two have commas before them which indicates that the sentence may be complex. (Note that the program was able to ignore the "and" in "candy" and the "but" in "butter".) Whatever the units the coordinating conjunctions do in fact join, they cannot be determined by the program without more help. At this point, then, CONTROL calls TALKCC and the "conversation" between student and system begins.

The samples above show only some of the features of the program's internal representation. As work goes on over time and new functions are added, that representation will grow increasingly rich. But it remains simply a preliminary to what one says to the student. Determining what to say and how to say it really means discovering what the student knows and applying at least parts of conversation theory. This is the direction in which the next chapter moves.

CHAPTER VI

STUDENT MODELS IN INTELLIGENT CAL

In the years since CAL ceased being what futurists talked about and became what instructional designers did, it has been quite clear that traditional branching strategies are not the stuff of educational revolutions. Since the Seventies, research has become far more oriented towards building intelligent programs that manipulate models of the real world as the student solves problems, rather than access frames as he answers multiple choice questions. At the same time, it also has become clear that a far better understanding of what the student knows at any one moment is crucial in developing intelligent instruction. This need exists not only from a theoretical point of view that a good regulator must have a model of the thing regulated, but from a growing number of practical experiences with CAL. Hartley (1978), for example, speaks of the "sparse" records kept of the student with the result that "the teaching is primitive in accommodating individual differences since the decision rules are largely based only on the last response of the student". Brown and Burton (1977) see the need "for a structural model of the student to direct the content and level of the tutor's comments". Self (1974) sees a model of the student at the center of "individualized teaching". Kearsely (1977) recognizes the power of such models to "diagnose deficiencies in learning strategies/tactics" but points out that this remains a "conceptual hurdle" for most current systems. As has been already discussed, Merrill et al. (1980) end their discussion by suggesting TICCIT's instructional model be expanded to account for various differences in students. Finally, my own work made it clear that even collecting mastery scores concept by concept in order to route the student to drill-and-practice made no significant difference. This chapter, therefore, examines some of the issues involved in building models of the student, looks at some of recent attempts to do so, and offers an account of a model used by COMCON.

As Kearsley (1977) has pointed out, "most current operational sytems have attained little more than personalization or response insensitive instruction". That is, there are not many programs that go beyond self-pacing despite the major claim of CAL to individualize learning. The reason for such a failure is fairly obvious: the five choices (and often less) that most CAL programs offer a student simply are insufficient for creating a sense of what he knows. A student can obviously generate much more variety than this, and one has to understand that variety in order to "regulate" or instruct him. Usually, CAL lessons can record the paths through right or wrong answers, but not the thinking processes which produced them. Simply finding out which are right or wrong answers is, of course, a straightforward task; trying to infer what the student might or might not have learned by interpreting those answers is far more difficult. Such inferences--when they can be made at all--are not easily done because the information likely is ambiguous or fragmentary. One finds, as Howe (1978) points out, systems that reason forward, looking for characteristics in the student's answer that would "imply a known mistake or 'bug' in his thinking". Or else, there are systems that work in the opposite direction to hypothesize a bug, insert it "in its pupil model and run the model as a simulation to see if it behavior accords with the pupil's behavior". But given the vast number of possible errors to anticipate, one is almost in the same situation as the writer of programmed instruction frames.

Self (1979) has cogently summarized many of the problems faced when one tries to model a student. Even before one can begin the process of buiding a student model, one must have chosen the proper representation of the subject matter itself. Since the representation problem in artificial intelligence and cognitive psychology is far from being solved, this presents obvious difficulties. The actual content of a student model is another central issue since one
cannot always be sure which are the essential features of every student's thinking. In addition, Self talks of the creation, change, growth, execution, comparison, planning,

monitoring, and the efficiency of such models. The list is formidable, to say the least. But as he insists, the whol subject of modeling the student is essential, not an "opt nal extra". Let us look at a number of prog ams which have in fact attempted to model the stu nt. Following Self, we should understand that the lit ature more often proposes models than implements the This review, therefore, at times reports schemes for uture work, not yet operational.

Very early CAL simply recorded right and wrong ans rs. Improvements such as those by Peplinski (1970) use this information to generate more appropriate ques ions, a tactic still employed. Similar facilities were supplied by Taylor (1968) and by Smallwood (1962). Ear Intelligent CAL (ICAL) work spoke of--but did not imp ment--more sophisticated models of the student. Cart nell (1970), whose lessons in geography used a "se ntic network" to represent its knowledge base, sug sted that "temporary tags" could be placed at key nod of the network to show how much of the network had beer accessed in respone to student questions. Looking at thes tags would be like looking at a history of the stud nt's interactions with the system. Wexler (1970) also used a semantic net in geography lessons and also sugg ted tagged nodes to see if they had been accessed. Self (1974) described a more dynamic "procedural student mode " used in lessons whose programs could predict ansv rs that would match the student's. He argued that all nowledge cannot be represented by a semantic net, and that further, the question of how the knowledge in a net is r lated *is* a procedural one. Stansfield (1974) also has uggested procedural models.

By the late Seventies, CAL designers had made con derable progress. O'Shea's (1979) "self-improving qua atic tutor" automatically amends teaching to fit hypc heses about what the student knows. His overall str egy is to present examples that increase the lik lhood of the student discovering the different rules nee d to solve quadratic equations. An "hypotheses test r" makes estimates of the student's current knowledge stat by considering how many and in what combinations the stuc nt has mastered the subject's three main rules. Thes become the left hand (or antecedent side) of "IF- HEN" production rules. In other words, they are con tions which if fulfilled will cause the system to alt its teaching strategy. The production rules func ion first to get the student to master a particular math matical rule, then a second (without losing control of he first), and so on until he shows complete mastery. But omplicating all this is that a student often will appe r to have mastered a rule yet still fail to apply it wher he should. He may also forget a rule, make lucky gues es, or make guesses that could be achieved by more thar one rule. The program attempts to compensate for the unce tainty of knowing whether or not mastery has been achi ved by weighting a set of values for its hypotheses whic range from CERTAIN to DON'T KNOW to CERTAINLY NOT. With these values calculated, the system is able to make reas nable estimates of what rules the student knows and so ach him accordingly. O'Shea's research has shown tha there is a significant improvement in the time to comp ete lessons although not in student scores. The work sug sts that a model of the student that charts which rul· are known to him permits a finer-grained choice of exar les with a resultant increase in teaching eff' iency.*

Sleeman and Hendley (1979) discuss a "Problem-Solving Moni or" for students who are interpreting nuclear magr tic resonance spectra. Students using the system have already learned a relevant algorithm to solve prot ems in the domain. The program derives its student mod by analysing explanations the student enters about the algorithm which led him to enter the series of arg ents he did. It pays special attention to inc sistent and incomplete student explanations and can, in f ct, cope with a fair portion of what it receives. The authors follow Pask (1975) who suggests that having the tudent enter his own explanations requires more inv vement (and so is of more educational value) than sim y watching the system list what it would do given a

---- ------

* lthough this does not necessarily increase eff tiveness.

part cular problem. After having made his assertion, the stuc nt enters his explanation in natural language from whic the system abstracts formal language statements with whic to represent his understanding. A "domain algc ithm" works the same problem and its "trace" is comr red to the student's. A series of comments from the syst m to the student about his explanation follows.

The authors note that that the system is not yet ful' "natural". A major problem is that students often omit from their explanations things which they wrongly bel ve irrelevant. The authors do not report on how much is ept on file about a particular student--how typical, for xample, are his current errors. Such a facility woul obviously present a fuller picture than the app: ently "one-shot" mode that was described.

Goldstein (1979) describes a model derived from a "ger tic graph" (of the game of WUMPUS) whose nodes repr sent rules and whose links represent "evolutionary" (th is, increasingly sophisticated) skills. He suggests that the graph allows far better student modeling than othe formalisms which generally view the student's know edge only as a subset of an expert's. This subset is "enc uraged" to evolve in the direction of the expert's by not g what is missing and then intervening. A graph, howe er, can represent an evolving structure of knowledge whi permits tutoring decisions to follow the leads given by he student's reasoning processes, not the expert's (whi h might, of course, be quite different). In prac ice, the system chooses a skill on the "frontier" of the tudent model, one which it assumes the student is abo to acquire, given his current abilities.

But despite rejecting the strategy that sees "expert know edge" as a superset of the student's, the system does comp re student performance with expert performance, albe t with the following improvements: Rather than a sinc e master performer, there are a number of "players" of t e WUMPUS game, each with increasing skills that corr spond to the skill plateaus on the graph. As the stu nt makes a move, each player assesses it and suggests which skills the student is using. The decision as to what skill he does in fact use is a weighted summation of t se suggestions with the advanced players furthest away from the current level of the student given the least weight. This results in a conservative model of the student which decreases the chance of assuming he is far m e skilled than he really is. Goldstein cites three major advantages of the genetic graph. First, it provides a more refined structure of skills for the student model. Second, it allows one to predict what skills are to be acquired next. Third, the links of the graph give a model of the learning behavior of the student and hence an idea o which strategies to call on.

Koffman and Blount (1975) and Koffman and Perry (1976) use a student model to adjust system responses to a computer science student's level as he writes short machine-language programs. A "probabilistic grammar" calculates which of various rules are being used by a student. This is combined with a measure of the student's proficiency level that is updated after each interchange.

This level also determines how much feedback the student will get. The model records the current and most recent change in the proficiency level for each concept, when each was worked on, at what level of proficiency, and the number of times the student either rejected or accepted a concept generated by the system. The system selects concepts for the student to work on according to a scoring polynominal and algorithm which decides if his level has just dropped or is changing rapidly, if a concept has not been used or is at the lowest level, or if a concept has more prerequisites than other contenders (and so can be used as a kind of review).

In summary then, current "intelligent" CAL uses models both to generate problems at the appropriate levels and to alter teaching strategies. This has always been a goal of CAL, of course, but the models permit a finer selection of new problems or remediation. Details of the models themselves appear quite different despite the similarity of function; this can best be explained by differences in the structure of the subject matter. It does not seem likely that a single formalism--such as

Goldstein's genetic graph, for example-- can be used in every single application.* The model I propose here extends current work in a number of ways. First, it is the only CAL for English grammar that to my knowledge attempts any sort of model beyond a simple record of right and wrong answers. (And many programs in composition lack even that.) Second, the model attempts to work explicitly from an algorithm very much like that given to the student. This makes it simpler to see where he has gone wrong and to inform him in his own terms. Often, algorithms in other CAL programs are used only by the expert performer or taught to the student, not to build student models. Third, the model provides information both about the last sentence the student has entered and his punctuation of it. The former is used by the system

----

* The punctuation skills taught by COMCON, for instance, are coordinate ones, in which there is an accretion of skills, not an evolution. Like representations of subject matter, student models are, I think, domain specific.

to create its own internal representation of the sentence. The latter is used to decide what remediation to offer. This decision is reinforced by examining not only the punctuation for the last sentence entered but for all sentences he has written, both in current and previous sessions. This naturally permits selection of remediation to be more responsive to changing abilities. Finally, the matrix formalism with its notation has the potential to be generalized to other kinds of intelligent CAL that is algorithm-based.

I do not intend the model to reflect learning styles or attitudes. Instead, it records each sentence individually and then adds it to the image of previously entered sentences. Therefore, what is known about the student is his use or misuse of the comma in sentences of his own composition. The model certainly can be combined with other information about the student when one develops an overall instructional strategy. This will be clear as we examine the way the system uses the model.

Recall that COMCON takes as input any sentence a student might enter and forms an internal representation of its punctuation. This is done by searching the sentence, looking for certain features, noting their configuration, and then drawing inferences about where mistakes may have occured. Hence, one has two sorts of information: what COMCON (the expert) thinks about the sentence and the student's own notions about commas as reflected by his sentence and by his conversation with COMCON. Further, once the student enters more than one sentence, COMCON then has several pieces of information which together constitute a history of what the student knows or doesn't. This history, cast into a form comprehensible to the program, can provide COMCON with the stuff of an educational conversation.

The same logic which informs COMCON's comma algorithm goes into building the student model although, as we shall see, the formats must differ. COMCON (like other computer programs) both "knows that" and "knows how": it knows that a properly punctuated compound-complex sentence looks like this:

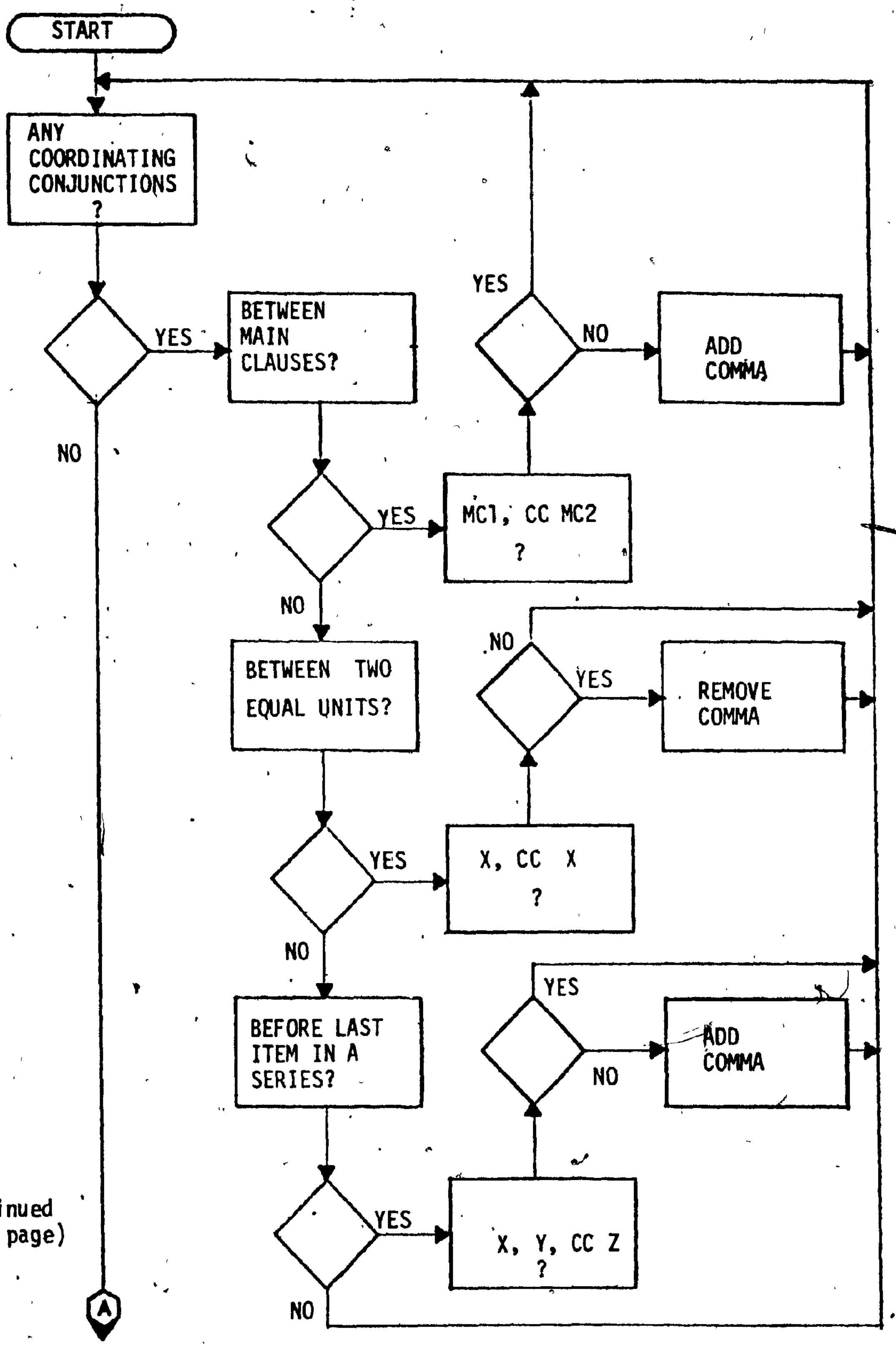
(I1),MC1(DC1),cc (I2),MC2(DC2) (,NE,)(S).

where I1 and I2 are introductory elements, MC1 and MC2 are main clauses, DC1 and DC2 are dependent clauses, cc is a coordinating conjunction, NE is a nonessential element, and S is a series. Knowing this, COMCON then knows what a properly-punctuated sentence looks like and how an input should be transformed. COMCON knows how to determine, for example, whether any particular element is a introductory element, a dependent clause, or any other component in the model. And to put that knowledge into practice, it follows an alogrithm to produce an internal representation of the input sentence.

A modified version of that algorithm has been used to teach students punctuation although the format the student sees is briefer from that which the computer uses.* Figure 5.1 shows the algorithm in flowchart form.

---

* In fact, the student first sees the algorithm in a fairly complete prose version, then a flowchart version, and finally as the acronym "CPINS". The increasingly shorter forms are intended to reflect the process of internalization by which the student frees himself from the complications of memorizing long sets of instructions.



(continued next page)

FIGURE 5.1A: The Comma Algorithm

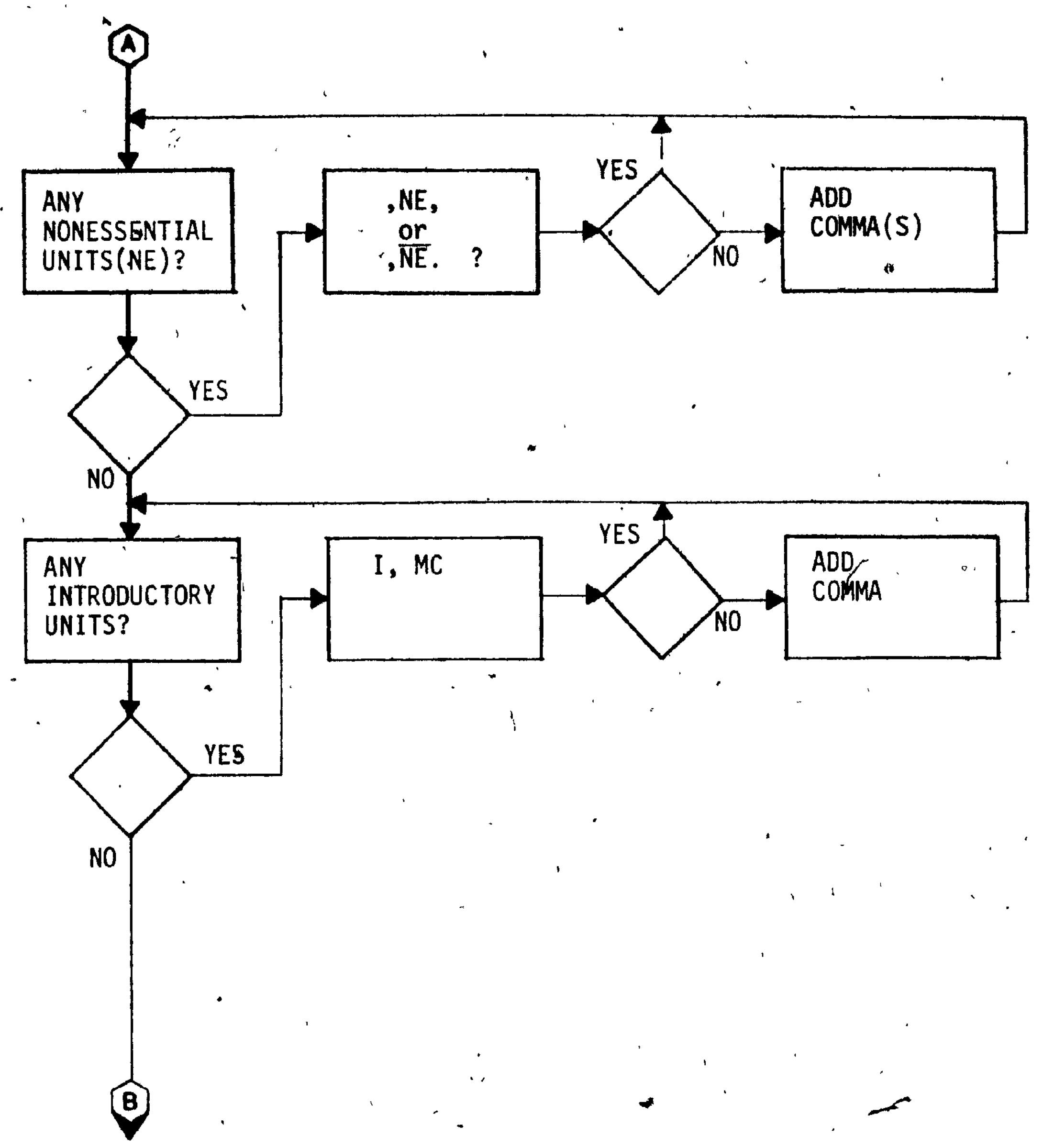


FIGURE 5.1B (continued)

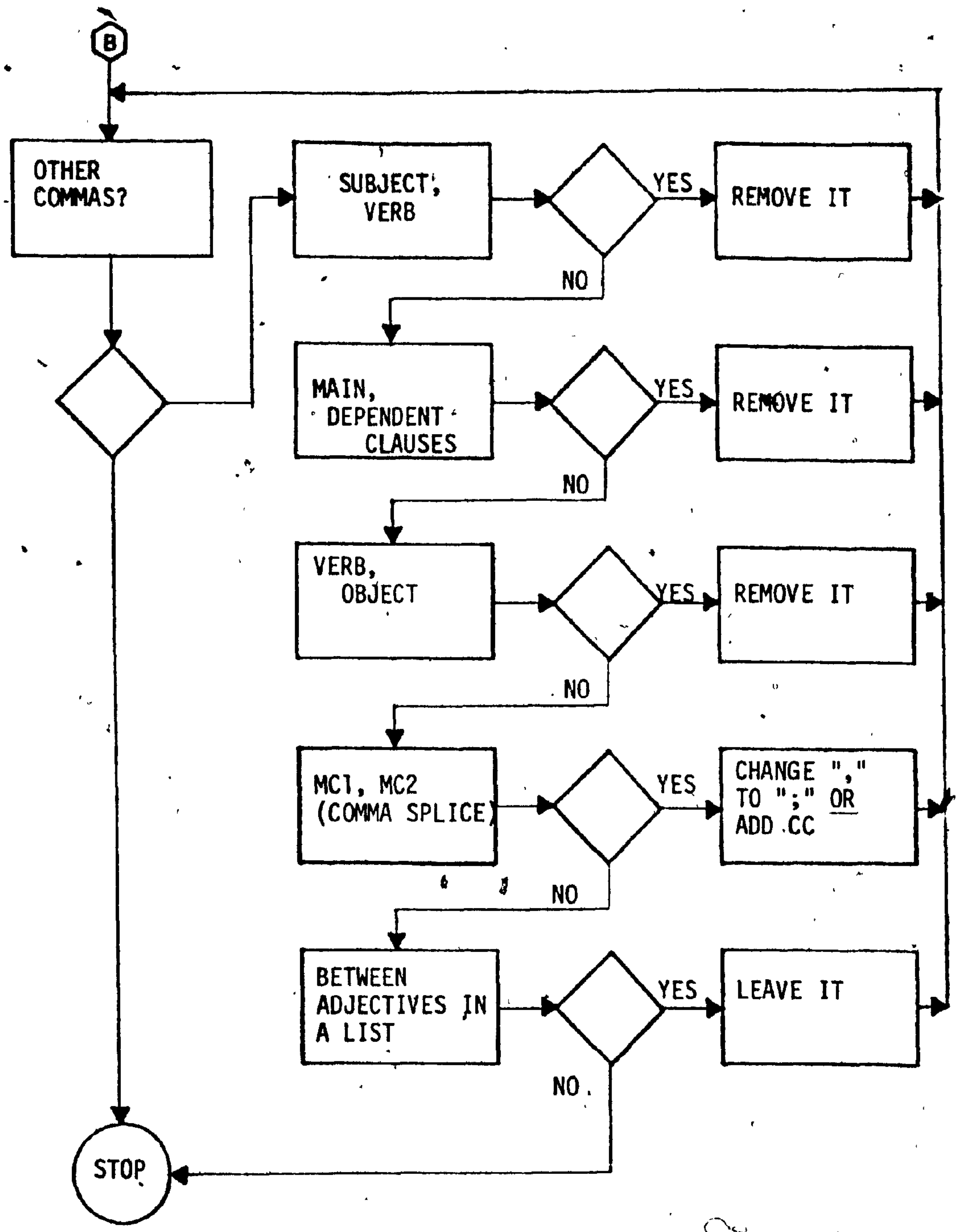


FIGURE 51C: (continued)

The algorithm directs the student to: perform a number of searches (for coordinating conjunctions, for dependent clauses, and so forth); to check to see that if such elements exist that they conform to the model of the properly punctuated sentence; and finally to make such changes as are required by the model. In effect, the student goes through the same procedures as COMCON but with a less explicit algorithm.

The algorithm itself has been shown to be successful in field tests (Keller, 1980). It therefore seems reasonable that one can create a student profile by searching for those places where the student, after having been taught the algorithm, uses or fails to it. For example, the algorithm says one must check to see if a coordinating conjunction joins the last element in a series; should the student not do this, we know something important about his understanding of punctuation. And this is so for each operation the algorithm specifies. Moreover, this information is routed to the conversation part of the program which then will call attention (or alert) the student to the feature in question and his punctuation of it.

The model itself is an n by 111 matrix where n equals the number of different sentences the student has entered over time. The 111 columns represent the various sentence features and the punctuation possibilities open to the student as he writes his sentences. The best way to think about the matrix is as a series of 111-element vectors, each of which images a single sentence input.

Each element of the vectors can have three possible values: 1, 0, and -1. A "1" signifies either that something is true (such as that there is a certain feature present) or that something is correct (a comma is in the right place). A "-1" shows that there is an error (a missing or an extraneous comma); an "0" shows that something is not there (a certain possible feature is absent) or, importantly, that there is no evidence on which to make a particular judgement. This last is central because although one would prefer to have used only Boolean values, there are times when it is impossible to say either "true or false". One must distinguish between what is false and what is merely unknown or undecided. There will be times when a student sentence simply will not offer any evidence at all if he knows a particular concept.*

A seemingly inescapable fact of producing student models is that they will often be incomplete. Figure 5.2 is a typical matrix (truncated here for convenience) that shows the data representing three sentences that the student has entered:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 1 0 1 1 0 1 0 0 0 0 -1
0 1 0 1 0 1 1 0 1 1 1 0 0 0 -1
```

Figure 5.2: Truncated Matrix Showing The Data Representing Three Student Sentences

----

* For instance, if his sentence does not have an introductory element, there is no way of knowing whether he does in fact know the rule about introductory elements.

The rows represent separate sentences; taken together as the matrix, they show a history of the student's exchanges with the system. (The values here are completely arbitrary; I include them only to provide an idea what the model looks like.) Figure 5.2 can best be understood by considering it as a series of vectors, each of which images one sentence the student has entered. Indeed, these vectors themselves are made up of a concatenated series of smaller vectors. Each element reflects either a feature in the sentence or an action that could be taken by following the punctuation algorithm.

To take one example, consider the vector that shows introductory elements. It must be more explicit than the comma algorithm in Figure 5.1 since it not only considers the whole introductory structure (that is, I, MC) but also the particular features that determine if a string at the beginning of the sentence is or is not introductory. In addition to this, the model must also show the student's actions.

To continue, this introductory vector has eight elements, the first of which (V1) says whether or not there is an introductory element in the input sentence. Its value, therefore, can be only 1 or 0 for yes or no (-1, recall, is reserved for when the student errs). V1 is assigned a value both by asking the student and by looking at the first word in the string for indicators of an introductory element. Should the student say there is an introductory element, COMCON scans ahead to see if a comma is present that might separate it from the main part of the sentence. If it does find a comma, COMCON asks the student if the function of that comma is to separate introductory and main parts. The system uses this answer to fill the second element (V2) which indicates if there is a comma after the intoductory element. The value here can be either 0 or -1.*

Should the student have answered, however, that there is no introductory element, the system would check that against V3 through V8 which are assigned by the system itself as it analyzes the sentence and to see if the first word in the string is a prepostion (V3), a subordinating conjunction (V4), a conjunctive adverb (V5), a relative pronoun (V6), a stock phrase (V7), or a single word interjection (V8).** The features in these last elements are important since all of them indicate that the main subject does not come at the beginning of the sentence and thus the sentence must include an introductory element. The combination of an "0" in V1 and a "1" in any of V3

----

* Note also that if V1 is 0 (that is, there is no introductory element), V2 must also be 0 to show there is no comma.

** In the case of the stock phrase (e.g., "As a matter of fact"), the scan naturally includes more than the first word.

through V8, therefore, indicates that the student has an introductory element in his sentence but has failed to recognize it. The following vector, for example, shows the case of a student whose sentence began with a subordinating conjunction but who did not recognize that this meant there was an introductory element which required a comma:

0 0 0 1 0 0 0 . . .

Using this information, the system can take the appropriate remedial action.

The same logic is used for the other subvectors, and there is one for each major section of the algorithm: coordinating conjunctions, nonessential units, other commas present in the sentence, as well as introductory units. The coordinating conjunction vector shows coordinating conjunctions between main clauses, equal units (further divided into single words and groups), and series (again divided into single words and groups). The "nonessential unit" subvector shows whether pairs of commas enclose the string (one or more words) in question. The string itself is determined by a conversation with the student which both asks him if something is nonesssential and alerts him about possible errors in its punctuation. And finally, the "Other Comma" subvector shows that commas not accounted for by any other concept are there because they either separate subject from verb, verb from object, main clause from following dependent clause, main clause from main clause, or modifier from noun.

Having completed a conversation with the student about his sentence, COMCON then concatenates all subvectors and adds them as the bottom row to the matrix which serves as the student model. The system also records the original sentence and a corrected version (which is generated as the conversation progresses and which is shown to the student for comparison with his own). COMCON now has a picture of the student that extends over time and it can check either just the last sentence or any previous ones.

In order to decide on remediation, COMCON examines the model in the following manner: A scan begins by looking for -1's in the bottom row (i.e., the last sentence). Having found one, it then goes up that column checking to see whether the error was habitual or a temporary lapse.*

Before the student enters his next sentence, the system will offer him a brief remediation on the topic. This is repeated for each -1 located. The scan then looks for all 1's which indicate correct punctuation. Again, the particular column is scanned, this time to see if the student has corrected previous errors or shows a history of being right. A comment is then selected from a store of readied ones. A third scan looks at the second to last row (the sentence previous to the one just entered) and again looks for -1's. If it finds one, it looks down to the last row to see if there that would indicate whether

----

* "Habitual" is defined as two or more errors in the last three attempts; "temporary" is defined as an error only in the last attempt.

that concept was not at issue in the latest sentence. This shows that the student erred but did not really attempt to correct himself. The system then presents an appropriate remark.

In fact, a matrix representation can accommodate a very large number of remedial strategies. One can scan beyond the last three sentences if one wishes; change the definitions of "habitual" or "temporary"; look not at single kind of error but at clusters of errors; and one can decide if the student needs remediation as brief as a single reminder or as long as a mini-lesson. The decisions will naturally reflect whatever instructional theory the teaching uses, but the point here is that a matrix representation allows one a sense of the student that is both current and evolutionary, and one can examine as much or as little of it as desired.

The model can be faulted in a number of ways. Unlike Goldstein's (1979), it does not permit one to predict which concept the student will need next. But this has more to do the subject matter than the model because the skills taught here are coordinate ones. That is, the student must know the use of the comma for all concepts equally well. Learning _each_ particular concept is, however, a hierarchal process: to know that a following subordinating conjunction should be separated from its main clause requires first knowing what a clause is, then what are main and subordinate clauses, and then subordinating conjunctions.

However, the model does have some predictive powers. If a student has consistently made a certain kind of error, the chances are that he will continue to do so. In such a case, generative CAL would produce examples to elicit that same error until it no longer existed. But COMCON does not generate examples; it assumes that the sentence the student writes is a reasonably accurate reflection of his skills. The model, therefore, makes predictions not to write new examples but to call up the appropriate level of remediation. The designer can specify any set of parameters he wishes in order to do this. Future research will consider, however, if generating examples (or at least displaying "canned" ones) might be "the appropriate level of remediation" in certain cases.

Another criticism of the model one might make, again following Goldstein, is that it represents a subset of the subordinating conjunctions.

However, the model does have some predictive powers. If a student has consistently made a certain kind of error, the chances are that he will continue to do so. In such a case, generative CAL would produce examples to elicit that same error until it no longer existed. But COMCON does not generate examples; it assumes that the sentence the student writes is an reasonably accurate reflection of his skills. The model, therefore, makes predictions not to write new examples but to call up the appropriate level of remediation. The designer can specify any set of parameters he wishes in order to do this. Future research will consider, however, if generating examples (or at least displaying "canned" ones) might be "the appropriate level of remediation" in certain cases.

Another criticism of the model one might make, again following Goldstein, is that it represents a subset of the master performer, not the student or, more precisely, the distance between what the student does and what the algorithm prescribes. This is a criticism common to all expert CAL systems which measure the discrepancy between what the master performer can do and what the student in fact does. The result is a model which may not adequately reflect the student's misconceptions. This is a theoretical criticism, but not much help at a practical level. If one could predict every possible misconception a student might have, one could simply write a frame for each one. But that is not feasible, even for a restricted but still realistic number. Instead, this program tries to infer from the input sentences and from conversations with the student where he failed to apply the algorithm he was taught. Remediation proceeds from there.

One might also add that a matrix with 111 or so columns and many rows is too big to be conveniently handled. It is here, I think, that using the computer is really justified. Without going into programming details, COMCON is written in APL, a language that manipulates matrices with great efficiency. Scans up, down, and across a matrix are primitive operations in APL. Functions are easily written to access any portion of the matrix according to the instructional strategy required, and these can be documented so that nonprogrammers can understand and use them. In fact, unless such difficult and complex computations were involved, what would be the point of using a computer in the first place?

Research shows giving a student an algorithm can be a means of effective teaching (see, for example, Landa, 1976; Gerlach, Reiser, and Breck, (1975)); it follows that algorithm-based student models should also be successful because they should make explicit where the student has gone wrong. However, we have to be able to recognize at which crucial points--and why--model and algorithm diverge. The work of Donald Norman (1979) is useful in discussing both algorithms and models. Norman speaks of "schemas" or "frameworks on which to interpret one's experiences".* An algorithm is like a schema because once an appropriate algorithm has been selected, information in the environment can be processed. The very strength of the algorithm is that it does not have to deal with only a single set of data but--theoretically--can deal with an infinite subvariety which falls within its classifications.

Norman also makes an important distinction between

----
* Such frameworks permit us to reduce the amount of processing of information we must do to make sense of the world and can also permit us to make inferences about things we cannot immediately perceive. Once we have selected the right schema for a particular set of experiences, we can deal with a wide variety of information, "imperfect matches" between our schema and our raw experience.

"conceptually driven" and "data driven" information. In the former, the concepts underlying a particular set of strategies are used as the system solves whatever problem it has, and deals only with relevant portions of data, ignoring everything else. In data-driven analysis, the opposite is true and the system works by examining pieces of data to see if they are important enough to attend to. (Assuming, naturally, that we can specifiy what is important.) The two procedures, although quite different in direction, can be worked together so that "one is capable of focusing . . . attention upon particular tasks yet not missing what is present in the environment."

This is a fair description of what one does when building an algorithm-based model. The model must reflect the probable data --what could be in the sentence-- and also the controlling concept for dealing with the data--what the expert and the student think must be done to the sentence. It isn't enough for the model to show only the actions the student takes; it must show what appropriate actions he could take potentially. Similarly, there must be a way of showing the actions of the expert, both actual and possible. A model, therefore, must contain information about the problem (and its possible permutations, combinations, and solutions), about the expert's manipulation of the problem, and, of course, about the student's. All three must be represented if one is to have a conversation with the student about his sentence's punctuation. An algorithm may have such information, but implicitly; a model must make it explicit. As a practical matter, when one says that the model must be more explicit than the algorithm given to the students, one means it is the most complete version of the algorithm. Second, a model must also show not only which paths have or have not been accessed, but paths about which there is insufficient information to make judgements about the student's ability. The notation used must distinguish among them. And third, because not every node in the algorithm reflects the same type of conditions or actions, there must be a distinction between what is observed about the sentence (is there a comma?) and what is done to it (was one inserted?).

I think this third is especially important because an algorithm is a kind of production system* --a pair of antecedent-consequent statements which direct the user to do something if certain conditions are met. These conditions are themselves the product of an "atrribute analysis" which determines what are the attributes of some particular object. For example, some (but not all) of the attributes of a properly-punctuated compound/complex sentence are the following: a comma between the two main clauses if they are joined by a coordinating conjunction and no comma between the main clauses and the following dependent clauses. This can be further broken down by considering the attributes of, for

----

* The term "production system" is closely identified with Newell and Simon 1972). More recent literature, for example, Waterman (1978), sees the production system as one kind of "pattern-directed inference system", that is, systems whose various parts may be activated by encountering different portions of input. See Chapter III.

example, a dependent clause: it begins with a coordinating conjunction, has a subject and a verb, and cannot stand alone and make complete sense. And still further analysis of a subordinating conjunction would show that it is a word that is either "before" OR "because" OR "when" OR "if" . . . etcetera. Both the writer of the algorithm and the writer of the student model must do the same kind of analysis down to whatever can be assumed to be the most primitive element.

Having done such an analysis, both a model of the student and an algorithm can be constructed. But not every attribute need be explicitly represented in the algorithm. One can expect that the student does not need everything spelled out in great detail--indeed, that much direction would produce an overly-complex set of instructions, not likely to be internalized. All the algorithm need do is provide a "controlling concept" (to appeal to Norman's distinction) and let the model account for the potentialities of the data. In the matrix model, each primitive element is represented by a cell which, as we've seen, can have one of three possible values: a 1 signifies either something is true (its existence can be asserted) or that something is correct and so very much like a Boolean 1. However, the Boolean 0 must be broken into the other two values: a -1 to show the student was incorrect and a 0 when either we can assert something is absent or we don"t know. This "neutrality" of value follows the problem that much of our information about the student will be incomplete or ambiguous. A master key, of course, must be maintained to indicate which numbers relate to which attributes.

It is, of course, a long and complex job to do the kind of analysis required to produce such a model. But without such efforts, one loses one great power of the algorithm to make teaching explicit and so help us model the processes of the student's mind. Lacking such a model, claims about the "intelligence" of CAL surely must be tempered.

CHAPTER VI

COMCOM: THE COMMA CONVERSER

AN OVERVIEW OF THE PROGRAM

This chapter describes The COMMA CONVERSOR (hereafter called COMCON), a general, quasi-intelligent CAL program that teaches the use of the comma in English sentences. "General" means COMCON can deal with any sentence the student might write, not merely "canned" ones. Such sentences can be either simple, compound, or compound-complex (although the the present version cannot go beyond two main clauses). "Quasi-intelligent" means that COMCON cannot produce a perfectly, punctuated sentence by itself the way an intelligent mathematics program can solve a problem. COMCON depends on information the student can provide about the semantic nature of the sentence before it can reach a solution. However, as I argue, this exchange between human and machine is itself an important part of the student's learning.

Figure 6.1 provides an overview of COMCON and its seven major components:

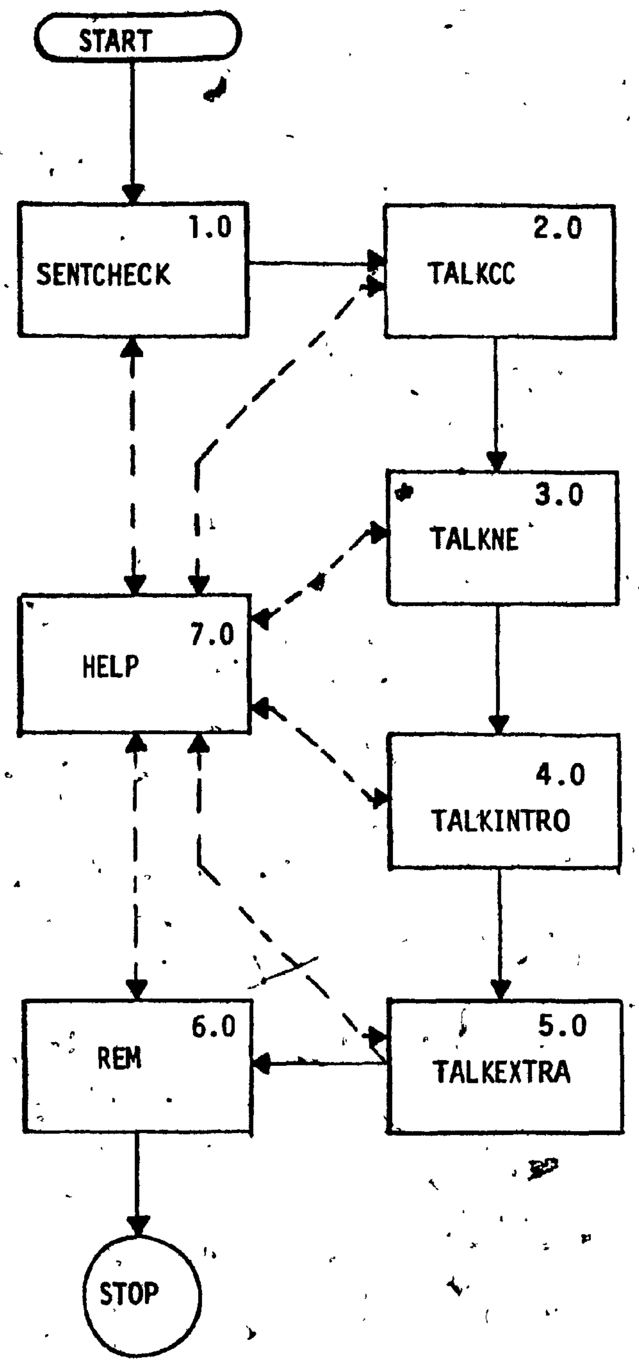
1) SENTCHECK, a "sentence checker" which accepts the input (any an English sentence) the student enters and builds an internal representation of its punctuation;

2) TALKCC, a set of conversations with the student about the sentence's coordinating conjunctions;

3) TALKNE, a set of conversations about its nonessential elements;

4) TALKINTRO, a set of conversations about its introductory elements;

5) TALKEXTRA, a set of conversations about any extraneous



FUGURE 6.1: THE COMMA CONVERSER
System Overview

commas;

6) HELP, a facility made up both simple information and interactive "mini-lessons" on terms and concepts encountered in the various "TALK" programs;

7) and REM, a program which looks at the information generated by the other programs and then produces a short review and remediation of student errors.

Each component is, in fact, a suite of APL functions using a total of about 400K of memory on an IBM 370. The large amount of space required makes it necessary to distribute COMCON over seven workspaces. This in turn means that COMCON must use the host computer's file system, not only to record student data, but to communicate key information about a sentence between APL workspaces. (This is much in the same manner as one uses APL's "global variables" to communicate information between functions in the same workspace.) Unfortunately, the amount of space and the number of scans COMCON must do make the cost of running the program in a time-sharing environment far too expensive for implemention into the present curriculum. Its use so far has been limited to relatively short sessions (about four or five sentences each) with a group of about ten students enrolled in a remedial writing course at Vanier College, an English-language CEGEP in Montreal. (CEGEP'S are roughly similar to junior colleges.)

After SENTCHECK has been loaded (either by the student or by the system), each of the TALK programs is automatically called in the order given in Figure 6.1.

The exception to this is HELP which the student can access at any time simply by entering the word "HELP". (This requires that each location where COMCON solicits input must be given a unique address which is used to return the student to the right place after he has done with HELP.)

Some sample runs of COMCON will provide a sense of its capabilities. The program accepts any sentence a student might enter and begins an exchange with him about the correctness of its punctuation. The sentences reproduced here have been taken from essays written by students in an Effective Writing course who had had about thirteen weeks of instruction in punctuation and grammar. This meant that although they were not expert writers, they were at least familiar with the terms and concepts they would encounter in COMCON. Unfamiliar items could be accessed through the HELP facility.

Once the sentence has been entered, most of the output below comes from the computer. However, this is what is needed to match the student's variety, not simply the generation of computer variety for its own sake. For the sake of clarity, asterisks show the student's words. The program begins with a few remarks and then invites the student to start:

```
ENTER A SENTENCE:

*IN  HIGH SCHOOL I DID NOT LEARN ANYTHING MORE COMPLICATED
THAN NOUNS, AND VERBS.

ARE THERE ANY COORDINATING CONJUNCTIONS IN YOUR SENTENCE?

*YES

READING FROM LEFT TO RIGHT, WHAT IS THE FIRST ONE YOU SEE?

*AND
```

RIGHT.

DOES IT JOIN TWO MAIN CLAUSES?

*NO

DOES IT JOIN TWO EQUAL THINGS?

*YES

ARE THEY SINGLE WORDS?

*YES

LET'S LOOK AT THE WHOLE THING.

WHAT IS THE FIRST WORD IN THE STRING?

*NOUNS

WHAT IS THE LAST WORD?

*VERBS

NOUNS, AND VERBS

ARE THEY JOINED ACCORDING TO "X CC X"?

*YES

NO, YOU'VE PUT A COMMA BEFORE THE COORDINATING CONJUNCTION.

REMOVE IT.

LIKE THIS:

IN HIGH SCHOOL I DID NOT LEARN ANYTHING MORE COMPLICATED THAN NOUNS AND VERBS.

ARE THERE ANY OTHER CCS?

*NO

THAT'S CORRECT.

IS THERE A NONESSENTIAL ELEMENT?

*NO

HERE IS YOUR SENTENCE:

IN HIGH SCHOOL I DID NOT LEARN ANYTHING MORE COMPLICATED THAN NOUNS AND VERBS.

DOES IT HAVE AN INTRODUCTORY ELEMENT?

*NO

WHAT ABOUT "IN"? A PREPOSITION AT THE BEGINNING OF A SENTENCE USUALLY MEANS THAT THERE IS AN INTRODUCTORY ELEMENT.

DO YOU THINK THAT THE GROUP OF WORDS BEGINNING WITH "IN" MIGHT BE INTRODUCTORY?

*YES

WHAT IS THE FIRST WORD IN THE STRING?

*IN

WHAT IS THE LAST WORD IN THE STRING?

*SCHOOL

HERE IS WHAT YOUR SENTENCE LOOKS LIKE WITH THE POSSIBLE INTRODUCTORY ELEMENT PLACED AT THE END:

I DID NOT LEARN ANYTHING MORE COMPLICATED THAN NOUNS AND

VERBS IN HIGH SCHOOL.

DO YOU THINK SO NOW?

*YES

YOU NEED A COMMA AFTER "IN HIGH SCHOOL".

ADD ONE--LIKE THIS:

IN HIGH SCHOOL, I DID NOT LEARN ANYTHING MORE COMPLICATED THAN NOUNS AND VERBS.

HERE IS THE CORRECTED VERSION OF YOUR SENTENCE:

IN HIGH SCHOOL, I DID NOT LEARN ANYTHING MORE COMPLICATED THAN NOUNS AND VERBS.

COMPARE IT TO YOUR ORIGINAL:

IN HIGH SCHOOL I DID NOT LEARN ANYTHING MORE COMPLICATED THAN NOUNS, AND VERBS.

"AND" JOINED TWO WORDS OF EQUAL IMPORTANCE, BUT YOU SEPARATED THEM WITH A COMMA. REMEMBER THE MODEL

X CC X

THERE SHOULD BE NO COMMA BETWEEN EQUAL THINGS.

REMEMBER, AN INTRODUCTORY ELEMENT MUST BE SET OFF FROM THE MAIN SUBJECT BY A COMMA.

YOU DIDN'T HAVE A COMMA AFTER THE INTRODUCTORY ELEMENT

THAT BEGAN WITH A PREPOSITION.

* * * * *

ENTER A SENTENCE:

*I STILL CANNOT HOWEVER, PUNCTUATE PERFECTLY, BUT AT LEAST, I NO LONGER PUT A COMMA WHEREVER IT SOUNDS CORRECT.

ARE THERE ANY COORDINATING CONJUNCTIONS IN YOUR SENTENCE?

*NO

YES, THERE ARE.

READING FROM LEFT TO RIGHT, WHAT IS THE FIRST?

*BUT

RIGHT.

DOES IT JOIN TWO MAIN CLAUSES?

*YES

LET'S LOOK AT THE FIRST ONE.

WHAT IS THE FIRST WORD IN THE STRING?

*I

THERE ARE 2 OCCURRENCES OF "I".   WHICH ONE DO YOU MEAN
(1, 2, 3, ETC.)?

*1

WHAT IS THE LAST WORD IN THE STRING?

*PERFECTLY

I STILL CAN HOWEVER, PUNCTUATE PERFECTLY

AND NOW THE SECOND--

START WITH THE WORD RIGHT AFTER THE COORDINATING CONJUNCTION.

WHAT IS THE FIRST WORD IN THE STRING?

*AT

WHAT IS THE LAST WORD IN THE STRING?

*GOOD

SO WHAT YOU ARE JOINING ARE:

I STILL CANNOT HOWEVER, PUNCTUATE PERFECTLY

AND

AT LEAST, I NO LONGER PUT A COMMA WHEREVER IT SOUNDS CORRECT.

ARE THEY JOINED ACCORDING TO "MC1, CC MC2"?

*YES

THEN YOU'RE RIGHT.

ARE THERE ANY OTHER COORDINATING CONJUNCTIONS?

*NO

THAT'S RIGHT.

IS THERE A NONESSENTIAL ELEMENT?

*NO

WHAT ABOUT "HOWEVER"?

HERE IS YOUR SENTENCE WITH IT REMOVED:

I STILL CANNOT ,PUNCTUATE PERFECTLY, BUT AT LEAST, I NO LONGER PUT A COMMA WHERE IT SOUNDS CORRECT.

DO YOU THINK NOW THAT IT'S NONESSENTIAL?

*YES

YOU ARE MISSING A COMMA BEFORE "HOWEVER".

ADD ONE--LIKE THIS:

I STILL CANNOT, HOWEVER, PUNCTUATE PERFECTLY, BUT AT LEAST

I NO LONGER PUT A COMMA WHERVER IT SOUNDS CORRECT.

YOU HAVE A COMMA AFTER "HOWEVER" WHICH IS CORRECT.

ARE THERE ANY OTHER NONESSSENTIAL ELEMENTS?

*NO

HERE IS YOU FIRST MAIN CLAUSE:

I STILL CANNOT, HOWEVER, PUNCTUATE PERFECTLY

DOES IT HAVE AN INTRODUCTORY ELEMENT?

*NO

LET'S LOOK AT THE SECOND MAIN CLAUSE IN YOUR SENTENCE:

AT LEAST I NO LONGER PUT A COMMA WHEREVER IT SOUNDS CORRECT.

IS THERE AN INTRODUCTORY ELEMENT?

*YES

IS IT MORE THAN ONE WORD LONG?

*YES

LET'S SEE WHAT IT IS.

WHAT IS THE FIRST WORD IN THE STRING?

*AT

WHAT IS THE LAST WORD IN THE STRING?

*LEAST

THE COMMA AFTER "LEAST" IS RIGHT.

HERE IS THE CORRECTED VERSION OF YOUR SENTENCE:

I STILL CANNOT, HOWEVER, PUNCTUATE PERFECTLY, BUT AT LEAST, I NO LONGER PUT IN A COMMA WHEREVER IT SOUNDS CORRECT.

COMPARE IT TO YOUR ORIGINAL:

I STILL CANNOT HOWEVER, PUNCTUATE PERFECTLY, BUT AT LEAST, I NO LONGER PUT IN A COMMA WHEREVER IT SOUNDS CORRECT.

THE MODEL FOR NONESSENTIAL ELEMENTS--

,NE,

--SAYS THERE MUST BE A PAIR OF COMMAS AROUND

A NONESSENTIAL ELEMENT.

YOU DID NOT IDENTIFY THE NONESSENTIAL ELEMENT AT FIRST.

YOU DID NOT HAVE A COMMA BEFORE "HOWEVER" WHICH WAS NONESSENTIAL.

* * * * *

ENTER A SENTENCE:

*THE ONE THING I HAVE LEARNED, IS THAT PROPER ENGLISH CAN

BE USEFUL, VERY USEFUL WHEN IT COMES TO IMPRESSING, PEOPLE YOU MEET.

ARE THERE ANY COORDINATING CONJUNCTIONS IN YOU SENTENCE?

*NO

RIGHT, THERE AREN'T ANY.

IS THERE A NONESSENTIAL ELEMENT?

*YES

IS IT MORE THAN ONE WORD LONG?

*YES

```
LET'S SEE WHAT IT IS.

WHAT IS THE FIRST WORD IN THE STRING?

*VERY

WHAT IS THE LAST WORD IN THE STRING?

*USEFUL

THERE ARE 2 OCCURRENCES OF "USEFUL". WHICH ONE DO YOU MEAN
(1, 2, 3, ETC.)?

*2

YOUR SENTENCE WITH THIS STRING REMOVED READS:


THE  ONE  THING I HAVE LEARNED, IS THAT PROPER ENGLISH CAN
BE USEFUL, WHEN IT COMES TO IMPRESSING, PEOPLE YOU MEET.
```

DO YOU STILL THINK THAT "VERY USEFUL"

IS NONESSENTIAL?

*YES

THERE IS A COMMA BEFORE "VERY USEFUL". THAT'S RIGHT.

THERE IS NO COMMA AFTER "VERY USEFUL". ADD ONE--LIKE THIS:

THE ONE THING I HAVE LEARNED, IS THAT PROPER ENGLISH CAN BE USEFUL, VERY USEFUL, WHEN IT COMES TO IMPRESSING, PEOPLE YOU MEET.

DOES IT HAVE AN INTRODUCTORY ELEMENT?

*NO

YOU HAVE A COMMA AFTER "LEARNED".

DOES IT SEPARATE

THE SUBJECT FROM THE VERB?

*YES

REMOVE IT--LIKE THIS:

THE ONE THING I HAVE LEARNED IS THAT PROPER ENGLISH CAN BE USEFUL, VERY USEFUL, WHEN IT COMES TO IMPRESSING, PEOPLE YOU MEET.

YOU HAVE A COMMA AFTER "IMPRESSING".

DOES IT SEPARATE

THE SUBJECT FROM THE VERB?

*NO

A MAIN CLAUSE FROM A FOLLOWING DEPENDENT CLAUSE?

*NO

A VERB AND ITS OBJECT?

*NO

TWO MAIN CLAUSES JOINED ONLY BY A COMMA?

*NO

TWO ADJECTIVES THAT DESCRIBE A NOUN OR PRONOUN?

*NO

SHOW WHERE YOU MIGHT PAUSE WHEN SPEAKING?

*YES

REMOVE IT--LIKE THIS:

THE ONE THING I HAVE LEARNED IS THAT PROPER ENGLISH CAN BE USEFUL, VERY USEFUL, WHEN IT COMES TO IMPRESSING PEOPLE YOU MEET.

HERE IS THE CORRECTED VERSION OF YOUR SENTENCE:

THE ONE THING I HAVE LEARNED IS THAT PROPER ENGLISH CAN BE USEFUL, VERY USEFUL, WHEN IT COMES TO IMPRESSING PEOPLE YOU MEET.

COMPARE IT TO YOUR ORIGINAL:

THE ONE THING I HAVE LEARNED, IS THAT PROPER ENGLISH CAN BE USEFUL, VERY USEFUL WHEN IT COMES TO IMPRESSING, PEOPLE YOU MEET.

THE MODEL FOR NONESSENTIAL ELEMENTS--

,NE,

--SAYS THERE MUST BE A PAIR OF COMMAS

AROUND A NONESSENTIAL ELEMENT.

YOU WERE MISSING A COMMA AROUND A NONESSENTIAL GROUP OF WORDS.

REMEMBER,

NEVER PUT IN A COMMA JUST BECAUSE

YOU MIGHT PAUSE WHEN SPEAKING THE SENTENCE.

YOU PUT INCORRECT COMMAS IN THE FOLLOWING PLACES:

THE COMMA AFTER "LEARNED"

SEPARATED THE SUBJECT FROM THE VERB.

THE COMMA AFTER "IMPRESSING"

INDICATED A SPEAKING PAUSE INSTEAD OF A

GRAMMATICAL SEPARATION BETWEEN LOGICAL ELEMENTS.

* * * *

Given a system of the kind described, what sort of instruction is possible? Recall that COMCON produces an internal represenation of a sentence that shows the presence, absence, and configuration of key function words. Recall too the model program produces a matrix that shows both COMCON's internal representation of the input sentence and a developing image of the student who entered it. How rich a "conversation" (to use a term explained below) can one have between system and student? The answer to that question, of course, will shape the overall design of the entire set of programs and the use to which they will be put. The purpose of this chapter is describe a set of programs which exploit the information about sentence and student, and so break free of from traditional branching CAL with its many limitations.

The term "conversation" and Conversation Theory (hereafter CT) in general will be of value here not because what follows is a full-scale attempt to apply the theory but because the work of Gordon Pask and his associates provides a powerful approach to examining an educational process. For Pask, learning results from structured conversations between two participants; these may be the traditional teacher and the traditional learner, two dimensions of the same learner (as when one "converses" with oneself), or, and obviously important for this work, between a learner and a teaching apparatus that has both a representation of the subject matter and a means of communicating with the student. The goal of a conversation is to reach an "understanding", and this can only be achieved by a demonstration of the student's repertoire of procedures such that an external observer could say they match the procedures of the teacher, be it human or otherwise.

A conversation, therefore, is a process by which the procedures of the participants are exteriorized. Only when a concept ceases to be merely an internal collection of, say, neural elements and becomes consciously related to a set of overt actions which can be performed can we claim that the concept has been understood. Hence, CT places great value on the learner's active and ongoing attempts to master the repertoire of the expert. For it is only as the learner makes public, if one will, what he can do that it can be tested. This testing, in turn, contributes both to the learner's confidence and to the teaching process itself.

The role of the teacher (or teacher and subject matter expert together) should extend beyond the time he participates in the conversation. Before instruction, he must create an "entailment structure" or map of the knowledge he wishes to teach such that it is coherent, consistent, and at least partly cyclic. This last is important since an entailment structure must "embrace" cyclicity: it must be possible to derive most topics from a number of starting points great enough to match the variety of the learner's entry level and conceptual schemas. The entailment structure provides theoretical statements of the subject and a set of analogies which link theory to examples drawn from the everyday world of concrete experience. Analogies are central to CT since they provide a means of going back and forth between the highly abstract (for example, probability theory) and the more concrete world (experimental designs and frequency counts). See Pask (1975). In addition to the entailment structure, the teacher must have some means of recording and controlling the sequence of learning and of seeing that understandings have, in fact, occurred. This role, in some cases, can be assumed by an appropriately programmed computer. Finally, the teacher must provide the full range of requisite learning materials (picture, tapes, explanations) which the student will use.

This in place, one is able to begin instruction which itself must consist of a "strict conversation". The main features of a strict conversation (in effect, the ground rules) are:

1) the participants converse only in the formalized language L;

2) the conversation is limited to topics found in the particular entailment structure;

3) topics "assimilated" by the student are said to be "understood"; that is, he can explain and derive a topic from others in the domain. This "systematic justification" which the student should be able to offer is crucial in determining whether understanding has taken place. Neither an explanation nor a derivation need be verbal: it can be a classifying or "model-building operation and is a satisfactory explanation insofar as the model can be executed in an external facility to bring about the formal relation underlying T" (that is, the topic). A nonverbal derivation is a learning strategy or a "concrete depiction of one or more topic derivations".

A key feature of instruction based on CT is that some means exists for permitting the student to demonstrate the procedures that constitute his "understanding" of any topic. Such understanding amounts to an agreement between student and teaching program about the categorization of and the distinctions made about the material. In a traditional teacher-student setting, the logistics of this are, naturally enough, simple: the teacher asks for a demonstration and the student provides it. (The ease with which it is accomplished does not mean, of course, that it is necessarily easy to evaluate the validity of the demonstration.) The exchange is done in natural language which has the obvious virtue of flexibilty if also the potential defect of ambiguity. However, in a CAL environment, the use of natural language is of necessity very limited.* The CASTE and INTUITION systems developed by Pask and his associates are means by which "strict conversations" can be carried out, and during the course of which, the student can exhibit his "understandings". A later and more powerful apparatus, THOUGHTSTICKER, enables such strict conversations to be far more flexible, far more capable of accommodating the quite legitimate differences among ways of approaching a particular subject. In all cases, however, it is essential that the instruction includes a means of the student "teaching back".

----

* See O'Shea's (1979) "self-improving" quadratic tutor as one means of dealing with the problem.

The summary provided here is but a mere outline: the considerable richness and complexity of CT and its attendant concerns has engendered a sizable literature. But at the risk of vast oversimplification, let me extract the salient features for my particular interests. Knowledge, first of all, is observable only as a set of procedures in the repertoire of some computing medium, human or machine. This means that instruction must provide a way for the student to demonstrate what is inside his head. This is intuitively true as well as empirically so.* The implication for CAL is that it is insufficient for a student to choose among a limited set of alternatives instead of the full range of possible interactions with the computer. Discriminating among various possibilities is but one kind of knowledge and it is legitimately tested by a multiple-choice exercise. But proof of other learning--any sort of composing, for instance--at the very least, requires something to come directly from the student, for him to initiate it. One

----

* Pask's work supplies a number of such studies.

wants to allow the student to choose not from merely a prescribed set of alternatives but from among his full range of internal possibilities. Again, it is his variety in which one is interested. COMCON therefore makes it a fundamental principle of its design that evaluation of the student proceeds as much as possible from samples of the student's sentences, not from his reponse to previously stored ones. This is, naturally enough, the central difficulty in building the system, but unless the system can categorize and respond to sentences of the student's invention, it cannot "know" if a topic skill --here, the composing of correctly punctuated sentences instead of distinguishing alternate forms of existing ones--has been learned. The failure to observe this distinction, it seems to me, accounts for most of the shortcomings of the English grammar CAL programs I have discussed. The workings of COMCON will be described in detail shortly;for now, I wish to assert that having the student enter his own sentences is roughly equivalent to having him manipulate a model of a properly-punctuated sentence. It does not meet Pask's criterion of demonstrating the derivation of a topic; but one can argue that a student who writes many properly-punctuated sentences is in fact in control of all topics in the domain of punctuation relevant to those sentences.*

Like CT, therefore, COMCON wishes to have the student exteriorize his mental processes. To that end, its central strategy is to ask a number of "what about" questions when it detects certain patterns in the sentence input. A "what about" question asks the student to consider some feature that he may have overlooked but which the system flags as potentially important. For instance, in the course of the conversation, the model could show that the student's sentence begins with a subordinating conjunction, but the student did not think

----

* Note that it is important to speak of several sentences since a single one may be correct by chance or at least be simple enough to present no problems. It is necessary, then, to collect data on several sentences to ensure an adequate model of the learner.

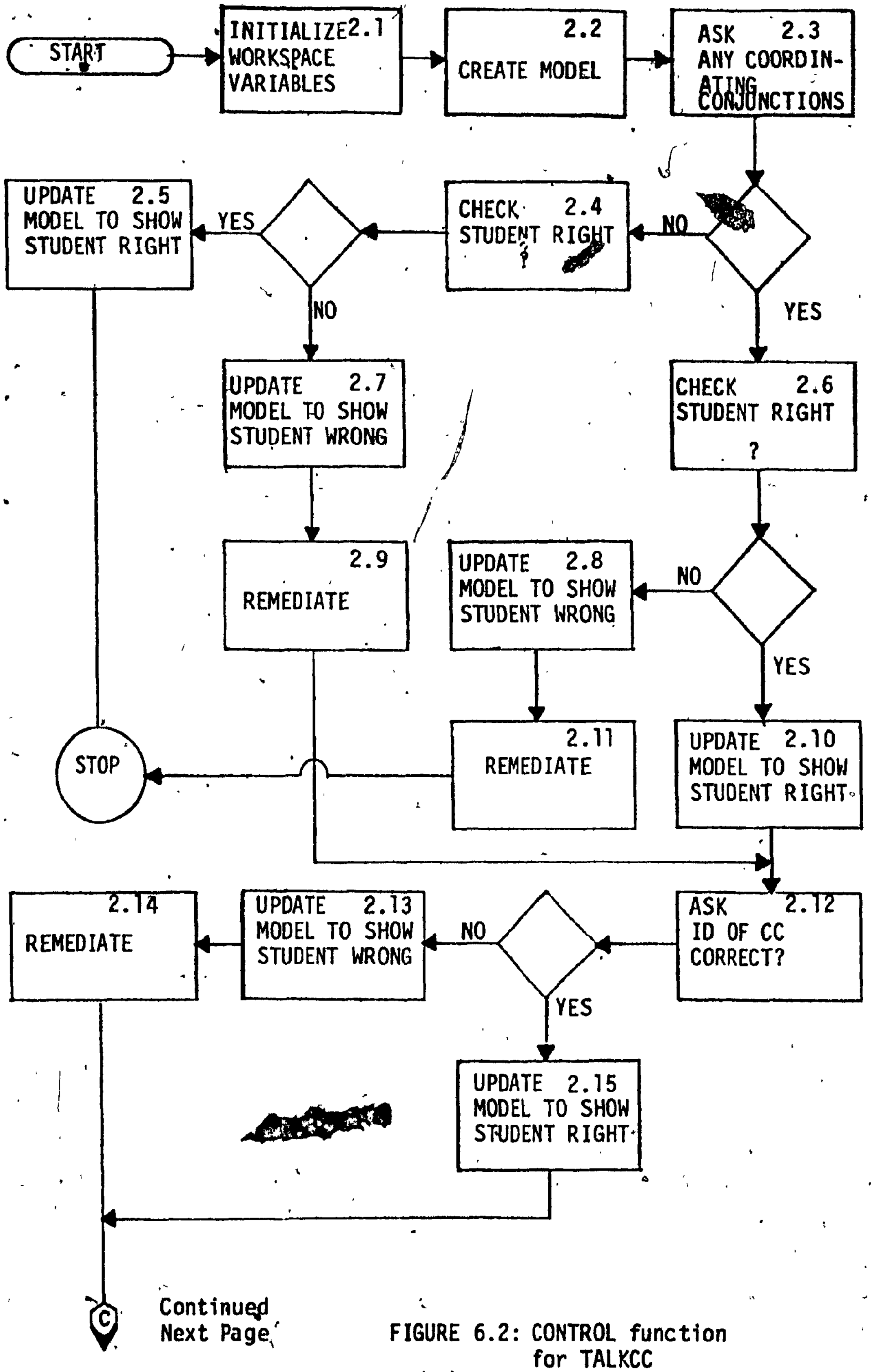
that there was introductory element. COMCON would then ask "what about X?" (the opening subordinating conjunction) and add that sentences begining in this way generally do have introductory elements. The student is therefore alerted to a possible error. In this sense, then, COMCON is a set of attention-directing procedures that cause the student to consider key features in his sentences that he might have overlooked. Note that the program cannot be more dogmatic than "what about" since it lacks a semantic processor of its own and so depends on the student. But the very asking of "what about?" an object causes the student either to consider or reconsider that object (the subordinating conjunction) and his treatment of it (if a comma is there or not). This is, I assert, a most powerful tutorial strategy, which even a human tutor could use. Further, the product of this asking "what about" leads the student to respond to COMCON which then uses that information to modify its own model of the sentence and to be able to ask still more focused questions. Hence, the exchanges sharpen the student's sense of his own work by drawing his attention to something he might have missed or did not know and benefits COMCON by reducing its uncertaintly about the semantic content and the structure of the sentence.

COMCON: The Main facility

This section describes some of the features of the facility that takes both SENTCHECK's internal representation of the input sentence and the model of the student and produces a dialogue about that sentence's commas. COMCON receives it basic information from SENTCHECK in the form of list of words that are associated with key features (coordinating and subordinating conjunctions, conjunctive adverbs, and so on); it also receives Boolean values relating to the configurations of these features (whether commas are present or absent before conjunctions, whether the first word in the sentence is a preposition, etcetera). The part of COMCON that converses with the student consists of four APL workspaces: TALKCC, TALKNE, TALKINTRO, and TALKEXTRA. The order that these programs are called derives from the algorithm outlined in Chapter III. This order is invariant because having the student consider whatever coordinating conjunctions are present in his sentence elicits important information about the structure of that sentence.* Figure 6.2 gives an overview of TALKCC.

----

* But note that this does not mean that every TALK workspace is called in its entirety. The nature of pattern-directed inference systems, as we have seen, is that they respond to data. This means that whole functions never appear unless the data warrants it.


START
INITIALIZE 2.1 WORKSPACE VARIABLES
2.2 CREATE MODEL
ASK 2.3 ANY COORDINATING CONJUNCTIONS
NO
YES
CHECK 2.4 STUDENT RIGHT ?
YES
NO
UPDATE 2.5 MODEL TO SHOW STUDENT RIGHT
CHECK 2.6 STUDENT RIGHT ?
UPDATE 2.7 MODEL TO SHOW STUDENT WRONG
NO
YES
UPDATE 2.8 MODEL TO SHOW STUDENT WRONG
2.9 REMEDIATE
STOP
2.11 REMEDIATE
UPDATE 2.10 MODEL TO SHOW STUDENT RIGHT
ASK 2.12 ID OF CC CORRECT?
NO
YES
UPDATE 2.13 MODEL TO SHOW STUDENT WRONG
2.14 REMEDIATE
UPDATE 2.15 MODEL TO SHOW STUDENT RIGHT
C

Continued
Next Page

FIGURE 6.2: CONTROL function for TALKCC

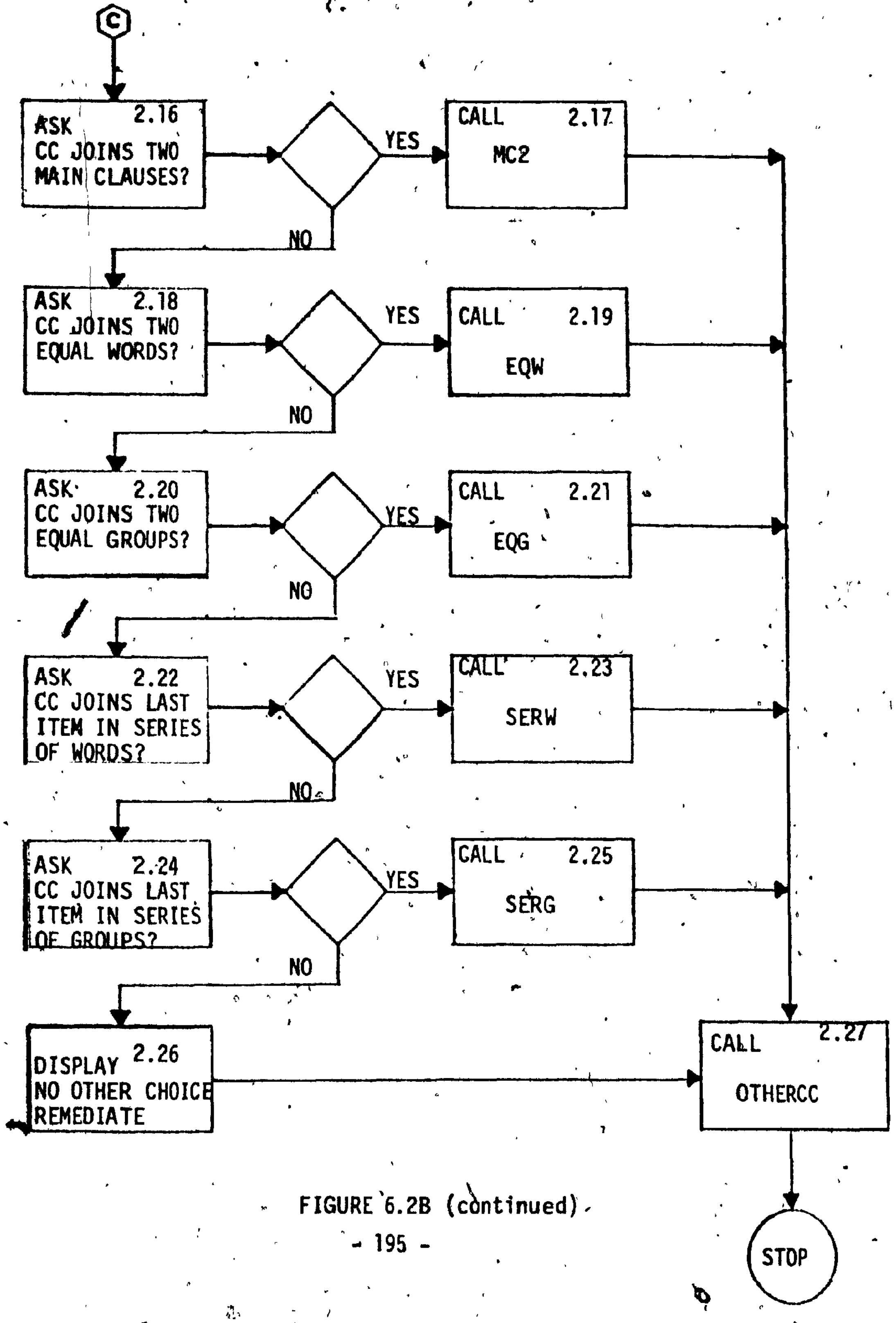


FIGURE 6.2B (continued)

The first question he is asked is whether his sentence contains any coordinating conjunctions. If it does, but the student responds with "No", TALKCC tells him that there is at least one and asks him to identify it. (Strictly speaking, this is not a "what-about" question, but it does serve the same purpose of directing attention to some pertinent feature.) Should he get it wrong, he is told what, in fact, it is. The exchange shows an external observer (actually, the model-building part of the system) whether the student can recognize a coordinating conjunction when he uses one, and the model is immediately updated to reflect this. At the end of the entire exchange, the system will have collected information about the student's knowledge of a variety of such features and can then begin appropriate remediation.

Once a coordinating conjunction has been identified, the system begins to take a closer look at the way it is punctuated. Because this requires some semantic knowledge, there is a list of enquiries: Does the coordinating conjunction join two coordinate things (like _running and jumping_ or _the tall blonde man and the short_ redhaired lady)? Does the coordinating conjunction join main clauses? Does it join the last element of a series? (Again, strictly speaking, these are not "what-about" questions but they have the same effect.) COMCON then questions the student to determine the length of these units.* Once it has this information, there is little problem for it to compare the sentence against the model of the properly-punctuated one. Should the student have the coordinating conjunction joining two main clauses without a comma in front of it, for instance, he is wrong and is told so. If the punctuation is correct, he is told that. The process continues until all the coordinating conjunctions in the sentence have been dealt with.

----

* Being able to recognize the boundaries of a unit is rather like a chunking operation. COMCON includes a number of heuristics to help the student identify the various units. For example, locating a dependent clause requires first locating a subordinating conjunction. There are several other similar heuristics.

At this point, one may quite legitimately ask what if the student either does not know what COMCON means (and cannot answer at all) or incorrectly thinks he knows (and gives a wrong answer). A HELP facility is available and can be accessed by the student at any time. So if he were to encounter the term "main clause" and not know what that meant, he could get help and then be returned to the right place in the COMCON program. There is, however, no simple answer to the problem of the student giving the system a wrong answer, but the thesis will try to address it below.

Before leaving TALKCC, COMCON makes a number of changes to its model. It notes which coordinating conjunctions the student has correctly identified and which have been correctly punctuated either before or after COMCON has directed his attention to them. It also presents the student with a corrected version of his sentence should that be required. (Typically, COMCON says "You are missing a comma before X. Add one like this", and then prints out the sentence with the new comma in place.) It also notes the presence of certain structural units which SENTCHECK could not uncover. For instance, SENTCHECK could see a coordinating conjunction easily enough, but not whether it joined two main clauses. This information is needed by TALKINTRO, as I will show shortly.

Having completed TALKCC, COMCON calls TALKNE which checks the punctuation of nonessential units (NE's). Figure 6.3 is a flow diagram of TALKNE. Here much depends on the student's ability to recognize elements of more than one word which may be nonessential. TALKNE, like the other programs, shares the information generated by SENTCHECK and so looks first at the list of conjunctive adverbs which, if not at the start of a main clause, often must be considered nonessential. Similarly, there is a list of stock parenthetical expressions (such as _it seems to me_, _indeed_, and so forth) for which SENTCHECK has searched. If the student says that his sentence contains no nonessential units, COMCON checks to see if one of the conjunctive adverbs or stock expressions has been located. "what-about" If none has, COMCON moves on. But if one does exist, COMCON asks a "what-about"

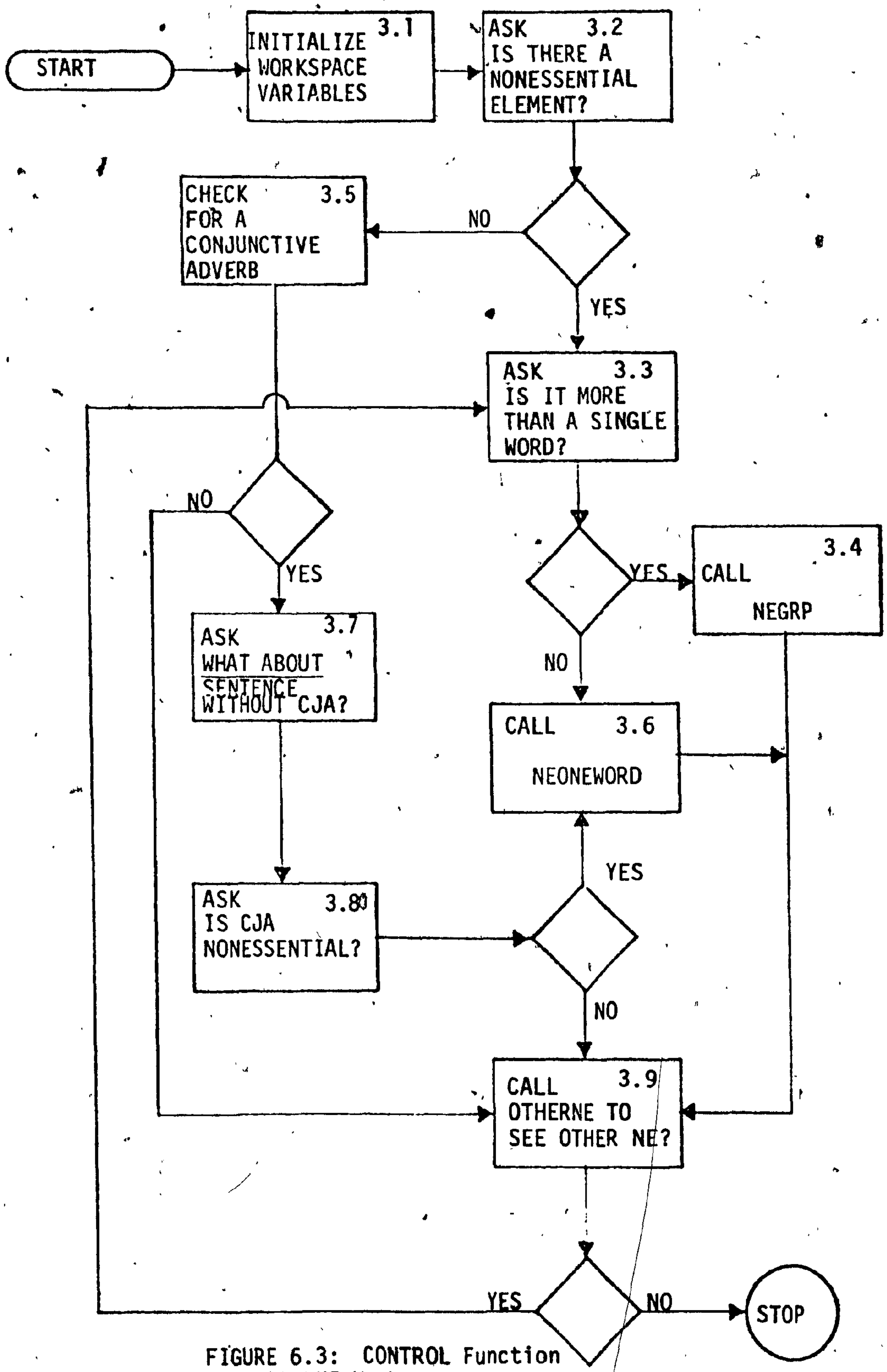

START
INITIALIZE WORKSPACE VARIABLES 3.1
ASK IS THERE A NONESSENTIAL ELEMENT? 3.2
NO
YES
CHECK FOR A CONJUNCTIVE ADVERB 3.5
ASK IS IT MORE THAN A SINGLE WORD? 3.3
NO
YES
YES
CALL NEGRP 3.4
NO
ASK WHAT ABOUT SENTENCE WITHOUT CJA? 3.7
CALL NEONEWORD 3.6
YES
ASK IS CJA NONESSENTIAL? 3.8
NO
CALL OTHERNE TO SEE OTHER NE? 3.9
YES
NO
STOP

FIGURE 6.3: CONTROL Function for TALKNE Workspace

question and follows this by removing from the sentence the word or words in question. It then says "Here is your sentence with "X" removed. Do you now think it's nonessential?" The test, of course, is whether or not the deletion of X destroys the meaning of the sentence (making X essential). But only the student can make that test; COMCON has no way of determining if the context of the sentence makes the element nonessential.* Again, one deals

----

* Consider the word "however" in the following two sentences:

However much you try, you will not succeed.

Even if you try, however, you will not succeed.

In the first case, "however" is adverbial and quite essential; in the second, it is a logical transition from the introductory dependent clause to the main one. And even the longer stock phrases can be tricky, as in this pair of examples:

with the uncertainty of how right or wrong the student is. But the alternative is to program for every possible contingency of the English language: this is neither practical, of course, nor possible.

Should the student say there is a nonessential element (NE), COMCON performs much the same procedure as before when it located a possible NE element. That is, it removes the string in question from the sentence and asks the student if he still believes it is nonessential. If the student changes his mind, the same procedure is repeated with other possible NE elements, whether they originate with the student or COMCON. If, however, the student maintains that the word in question is

----

I said, by the way, dinner's at eight.

By the way I said dinner's at eight, you could tell I was hungry.

nonessential, COMCON checks to make sure it is preceded and followed by commas. As before with TALKCC, COMCON reports errors or supplies confirmation to the student, shows him a corrected version of the sentence, and updates its model.

COMCON next calls TALKINTRO and checks what word begins the main clause (or clauses if TALKCC determined there was more than one). Figure 6.4 gives an overview of TALKINTRO. It looks specifically to see if the first word in the string belongs to one of four classes: prepositions, subordinating conjunctions, conjunctive adverbs, or stock phrases. (The current version does not yet have a dictionary of stock phrases, but one eventually will be written.) The presence of any one of these indicates that there is the very strong likelihood of an introductory element. In the case of a subordinating conjunction or preposition, it is a virtual certainty, the only exceptions being when an entire dependent clause or prepositional phrase itself constitutes the subject of a

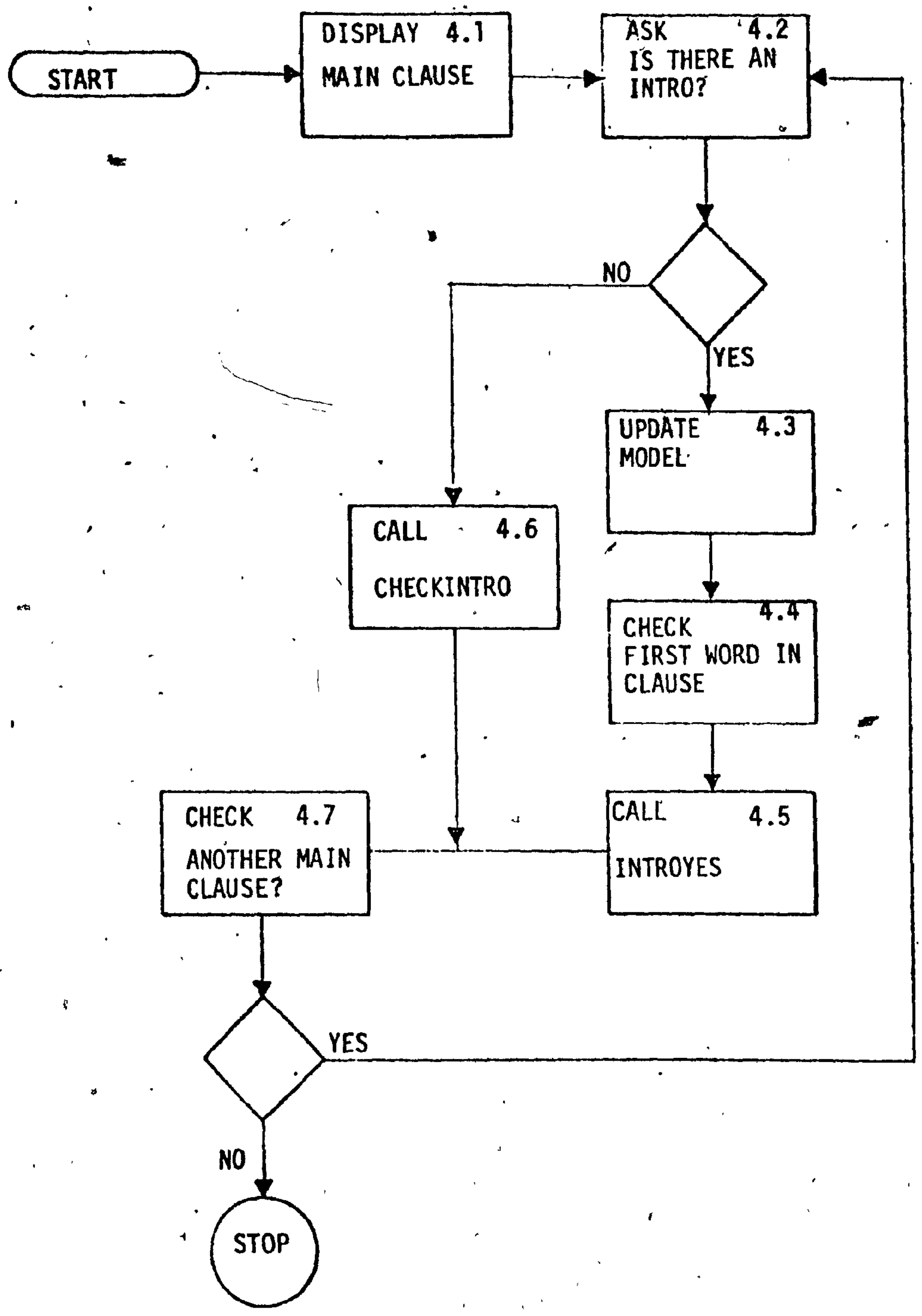


FIGURE 6.4 CONTROL Function for TALKINTRO Workspace

clause.* A subordinate conjunction in the first position, of course, signals a clause that can't stand by itself and so must precede the the sentence's subject which can occur only in a main clause. Similarily, the noun in a prepositional phrase is by definition the object of the preposition, and naturally, a word in the objective case cannot be the subject of anything. Conjunctive adverbs and stock phrases can be, as shown above, nonessentials; shifting them to the start of a clause changes only their structural function from NE to introductory, but does not solve the semantic problem. (Consider shifting "however" to the first position in "I think, however, you must do better").

As with the other "TALK" workspaces, TALKINTRO asks the student if he sees an introductory element and then checks that answer against the words it has taken from the sentence file. If one of those words does indeed appear

----

* As in, "'By the time I get to Phoenix' is the first line of a popular song."

and the student hasn't identified it, COMCON asks a "what about" question. This is followed by taking the word and shifting it to the end of the main clause. A true introductory element will still make sense because the sentence has been restored to the normal English "Subject-Predicate" word order.** If the student maintains that there is no introductory element, COMCON accepts this. Once more, we cannot program for every context. If, however, the "what-about" question results in the

----

** Compare shifting this pair of potential introductory elements:

By the way he talked, you would think he was angry.

he talked, you would think he was angry by the way.

If you try, you will succeed.

You will succeed if you try.

student asserting that there was an introductory element, COMCON makes corrections, shows the revised version, and updates its model. The same kind of test is done with potential introductory elements the student identifies, with the appropriate comments, corrections, and updates.

The last program COMCON calls is TALKEXTRA. Figures 6.5 and 6.6 give overviews of it and its main function, ASKEXTRACOMS.

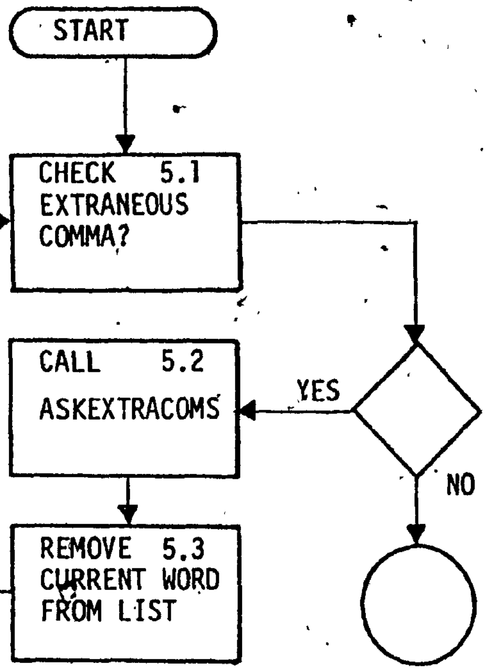

START
CHECK 5.1 EXTRANEOUS COMMA?
CALL 5.2 ASKEXTRACOMS
REMOVE 5.3 CURRENT WORD FROM LIST
YES
NO

FIGURE 6.5: CONTROL Function for TALKEXTRA Workspace

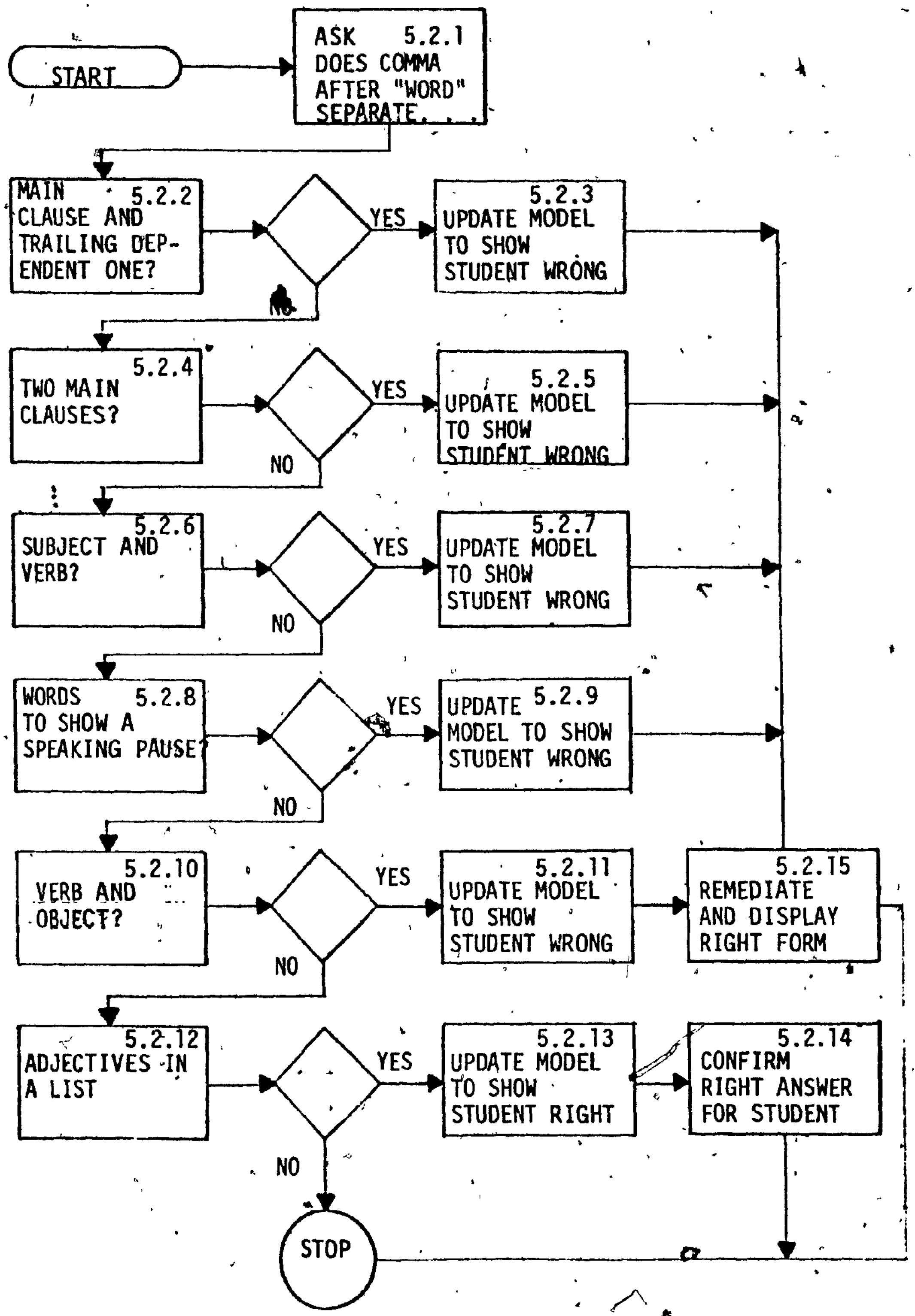

START
ASK 5.2.1 DOES COMMA AFTER "WORD" SEPARATE. . .
MAIN CLAUSE AND TRAILING DEP-ENDENT ONE? 5.2.2
YES
NO
5.2.3 UPDATE MODEL TO SHOW STUDENT WRONG
TWO MAIN CLAUSES? 5.2.4
YES
NO
5.2.5 UPDATE MODEL TO SHOW STUDENT WRONG
SUBJECT AND VERB? 5.2.6
YES
NO
5.2.7 UPDATE MODEL TO SHOW STUDENT WRONG
WORDS TO SHOW A SPEAKING PAUSE? 5.2.8
YES
NO
UPDATE 5.2.9 MODEL TO SHOW STUDENT WRONG
VERB AND OBJECT? 5.2.10
YES
NO
5.2.11 UPDATE MODEL TO SHOW STUDENT WRONG
5.2.15 REMEDIATE AND DISPLAY RIGHT FORM
ADJECTIVES IN A LIST 5.2.12
YES
NO
5.2.13 UPDATE MODEL TO SHOW STUDENT RIGHT
5.2.14 CONFIRM RIGHT ANSWER FOR STUDENT
STOP

FIGURE 6.6: Flow Pattern for ASKEXTRACOMS Function

To this point, COMCON's instruction has concentrated on missing commas; it now turns to the no less grievous problem of extraneous ones. Probably few things are more distracting (spelling mistakes aside) than commas liberally sprinkled on the page, forcing the reader to backtrack or guess which unit is which. Throughout the whole session, COMCON has maintained records about which commas in the original sentence were required by coordinating conjunctions, nonessential elements, and introductory elements. As each was accounted for, it was removed from the potential list that TALKEXTRA reads from the sentence file. TALKEXTRA asks "what-about" questions, therefore, for each comma not previously explained. Does it separate subject from verb? Verb from object? Main clause from following dependent clause? Two main clauses? Adjective from noun? Signal a speaking pause? With the exception of the separation of adjective from noun, none of these logically permits a comma. In some cases, such as when COMCON notes a comma between a main clause and a dependent one, the "what-about" question becomes more pointed since there could either be an error (the usual case) or a previously unacknowledged nonessential element at the end of the sentence.* The issue is again a semantic one that must be solved by the student. As each comma is accounted for, once more COMCON makes comments, corrections, and updates its model.

Finally, having completed all the analysis and "what-about" questions, COMCON presents the student with two versions of the sentence, the original and the latest corrected one. The student can compare the differences

----

* Consider this pair:

You will pass if you try.

You will pass, if you try.

The first shows the normal relationship between a main clause and a following dependent one; in the second sentence, however, "if you try" really is a parenthetical expression, an aferthought, that can be separated by a comma.

✓

between them and COMCON offers a brief review and appropriate comments about the errors that were made. The student then chooses one of four paths: 1) enter another sentence and have the process repeat anew; 2) use the HELP package; 3) use a set of introductory lessons; 4) exit. In the last case, of course, nothing is to be done except file away results for another session. The HELP facility and the introductory lessons will be described below. Here, consider the events that follow if the student opts for another round.

Before giving him a chance to enter his new sentence, COMCON checks its model for errors over two dimensions: the last sentence entered and the whole set of sentences entered at all previous sessions. Depending on the kind and frequency of errors it has located, it provides remediation. If the student has erred only on his last try, remediation can be as brief as a simple reminder to watch out for some previous pitfall. If, however, the error is more frequent, COMCON may suggest the student use the HELP package. (And to what extent. Different levels of HELP are available for every topic.) If the problem is severe, future versions of COMCON will not merely suggest but indeed will route the student to HELP before allowing him to continue. And in cases where there truly are problems, it will suggest or route the student to the introductory lessons. A separate set of records must be made of the advice given and taken.

In summary then, COMCON takes the information collected by SENTCHECK and adds the information it has gathered in its exchanges with the student. The typical form of these exchanges is a "what-about" question which directs the student's attention to features he might have missed or misjudged. As mistakes are found in the original, COMCON generates new versions of the sentence. And at the end of the process, COMCON gives a review consisting of original and new versions accompanied by the appropriate remedial comments. What I would like to emphasize, however, is that the entire process has centered around what the student has entered in the first place. The TALK functions respond to the learner's sentence and his ideas about it. And this is the basic design feature since the goal is to teach and test understanding of composition skills, not error recognition.

The HELP Facility

In the last section, I touched briefly on the problem of a student who lacks the basic knowledge required to respond to one of COMCON's questions. That is, if COMCON must know if a part of the input sentence is a main clause or a phrase, then the student must know that terminolgy if the entire process is to go forward. This problem likely can never be fully surmounted, but its damaging effects can be limited in a number of ways. The first is by the numerous heuristics built into COMCON that look for key features of English. If the student thinks, for instance, that his sentence does not begin with an introductory element (perhaps because he is not sure what one looks like), but if COMCON finds a preposition or a subordinating conjunction, then COMCON can alert him. I have outlined a number of such heuristics in previous sections. But once again, one faces the problems of natural language processing and the impossibility of dealing with sufficent context to analyze any and all sentences a student might enter.* A key design decision has been to use the student as a semantic processor on the assumption that, properly prepared and supported, he can provide semantic information to COMCON which in turn can supply him with information about punctuation. The HELP facility is one of the means COMCON has to "properly prepare and support".

HELP is a series of about twenty self-contained

----

* Not to dismiss the issue in a footnote, I recognize that a sentence fragment (such as Eating_fruit) would quite defeat COMCON in its present version. If a student insisted that a fragment was indeed a grammatically complete and meaningful sentence, there would be very little COMCON could do. It might say that normally such strings are not acceptable and that the "fragment" probably needs to be attached to another string. But it could not insist. Indeed, the current version remains silent in such a situation.

modules that deal with COMCON's key concepts. Figure 6.7 shows a typical flow pattern for a HELP module and Figure 6.8 shows the flow pattern for all of HELP's topics:

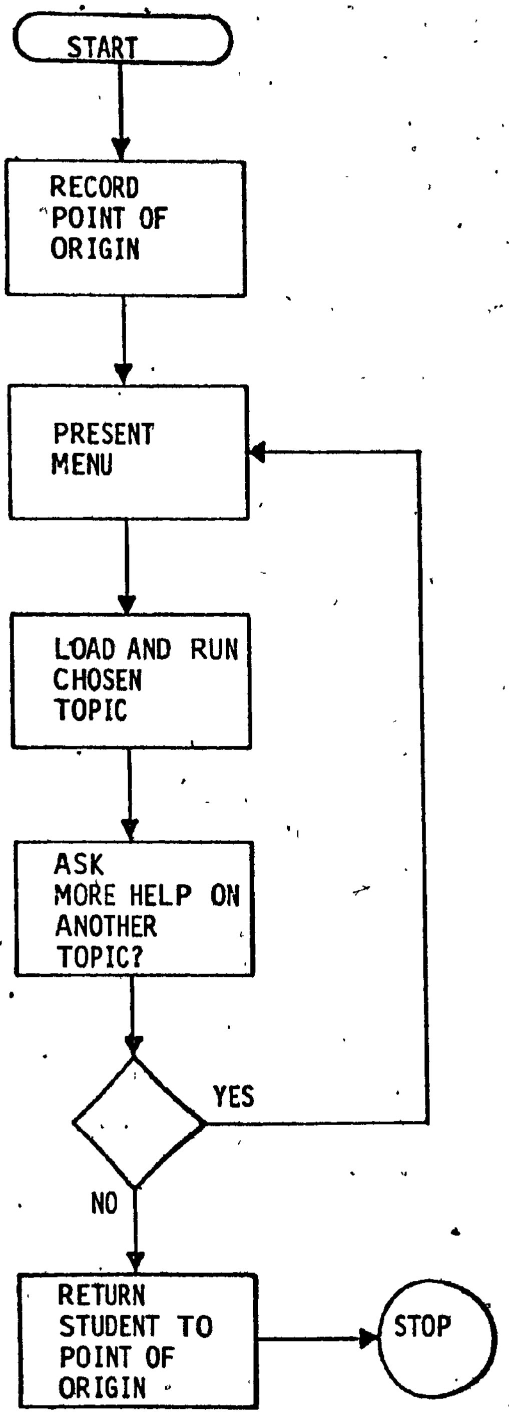

START
RECORD POINT OF ORIGIN
PRESENT MENU
LOAD AND RUN CHOSEN TOPIC
ASK MORE HELP ON ANOTHER TOPIC?
YES
NO
RETURN STUDENT TO POINT OF ORIGIN
STOP

FIGURE 6.7: Flow Pattern for HELP Workspace

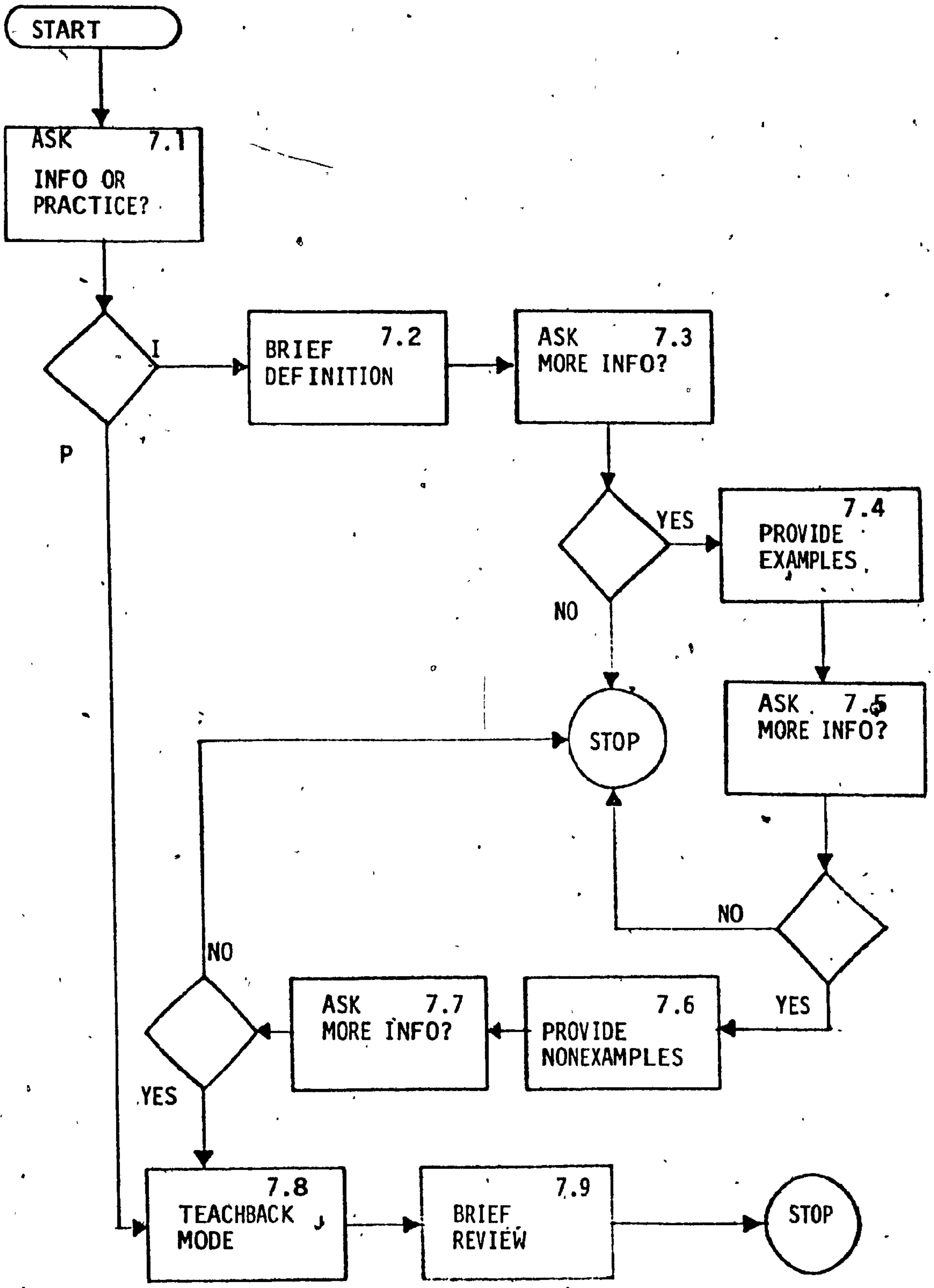


FIGURE 6.8: Flow Pattern for Each Topic in HELP

HELP can be accessed by the student whenever COMCON solicits input from him.* HELP begins by presenting a menu of topics and then loading the chosen module. Each module has the following components:

1) A brief (no more than a sentence or two) description or explanation of the topic;

2) An amplification of 1) with a set of examples and analogies wherever possible;

3) A set of counter-examples;

4) A practice session (which I discuss in more detail below);

---

* Each portion of the entire system is given a unique location code which is written into a file whenever HELP is accessed. After the student has finished, the system reads that file and automatically returns him to his point of origin.

5) and finally, a brief (about four or so lines) review.

The student always has the option of exiting as soon as he has received what he thinks is sufficient information. He also can go directly to a HELP practice section and repeat this as many times as he wants to. Again, as in the work of Pask, a key assumption is that the student must demonstrate his understanding in a "teachback" session. The sentence the student enters in response to the practice session's request is a concrete demonstration of the degree of his understanding and his ability to manipulate the model of the properly-punctuated sentence.

The request for a sentence in HELP generally includes certain specifications about that sentence. For instance, to simplify the matter, HELP limits input sentences to simple ones (in the strict grammatical sense), and so insists that there be no subordinating conjunctions that would indicate the presence of a dependent clause. One does face the problem that even in the HELP package a student will need help. The facility is not, however, recursive in its present version. For better or for worse, one must avoid an infinite regress of calls for help about HELP. One could always, of course, tell the student to seek his teacher's help, but this would change the self-contained nature of optimal CAL.

After the practice and review sessions, HELP asks if the student wishes to get help on another topic. It is perfectly legitimate, in fact, for the student to spend most of his time in HELP if he wishes since, taken together, these modules constitute a fairly complete treatment of the comma. Indeed, HELP has been designed as a learner-controlled alternative to the traditional CAL lessons also available on the system. The advantage to this is that the student can have it two ways: he can opt for HELP's modules--exploring, exiting, returning at will--or someone else (human or machine) can select a path through them. If an instructor were to choose system control, there would naturally have to be some rationale for the particular routes chosen for each individual student. Learning style is a possible candidate. The present version of HELP does not have such a facility since it lacks a way of gathering information about the student before he begins. It does have, however, a suggested path for students who want one but this is not enforced.

Figure 6.9 is a list of HELP modules available:

"HELP" is available for the following topics:

1. COORDINATING CONJUNCTION(CC)
2. SUBORDINATING CONJUNCTIONS
3. CLAUSES
4. MAIN CLAUSES
5. DEPENDENT CLAUSES
6. INTRODUCTORY ELEMENTS
7. NONESSENTIAL ELEMENTS
8. SERIES OF ITEMS
9. SUBJECT OF A SENTENCE
10. PREDICATE OF A SENTENCE
11. NOUNS
12. VERBS
13. MC1, CC MC2 (COMMAS IN CLAUSES)
14. X CC X (JOINING TWO EQUAL THINGS)
15. I, MC (COMMA AFTER AN INTRODUCTORY ELEMENT)
16. X,Y, CC Z (JOINING ITEMS IN A SERIES)

FIGURE 6.9: "HELP" MODULES AVAILABLE

It is not an entailment scheme in the rigourous use of the term. One can, I think, legitimately say that the topics are interrelated, but whether this means derivations are possible is uncertain. In essence, learner control is built into HELP so that the student can select the portions of it that he thinks (or has been told, in some instances) he needs. But he cannot at present absolutely derive one topic from any other.

Like CONCON, HELP's practice sessions try to analyze input rather than give canned sentences.* Each practice

---

* In fact, in the present version, there are about three or four modules that do use canned sentences. They do, however, randomize these canned sentences (each list typically has about ten or fifteen items) to reduce the chances of the same examples being repeated. Later versions of HELP will try to correct this problem which is caused because certain concepts--main clauses, for instance--cannot be fully analysed without more sophisticated semantic processing.

session begins with a request for a sentence (subject to certain limitations) and once HELP has satisfied itself that the sentence is one, it begins its analyis. So, for example, should the module on introductory elements be called, the analysis includes searching for prepositions, subordinating conjunctions, and commas. It then intitiates a dialogue about whether or not the sentence has an introductory element and, if so, whether it is correctly punctuated. In the module dealing with the subject of an input sentence, HELP asks the student for his notion of what the subject is, and then does a set of analyses including seeing if the word is part of a prepositional phrase (which means it cannot be the subject) or is preceded by a determiner (which would make it a noun and a good candidate). At that point, HELP suggests a number of tests to the student to see if the word could be the subject. One must recognize, however, that the final judgement is left to the student; there is no way for HELP, other than making a set of suggestions about tests, independently to determine whether the student is right or wrong. The student--whatever his deficiencies--is a far better semantic processor than HELP; so the system depends on him.

This last issue raises the larger one to which I have already alluded: what are the dangers inherent in having the student provide semantic information to the system? As was said, the HELP package itself tries to minimize the dangers by providing information and practice. In addition, one can pretest students to see if they have the necessary entry level to use the system, and if not, then provide it either in the classroom or by another set of programs. These two strategies are, in fact, part of the overall implementation scheme.* There are about twelve hours of the traditional CAL lessons on English grammar available (Keller, 1979). However, as we have seen, the

----

* One might also have students work in pairs. Although this could reduce the amount of wrong information being reported back, it weakens the emphasis on the individual learner. And two poor students may only reinforce each other's errors.

present set of programs grew out of dissatisfaction with these lessons because they failed to produce a transference of skills. The original lessons, however, did work well within their limitations and can be an introduction to terminology along with HELP. These lessons can, in cases where students have persistent difficulty, be used to ensure at least some familiarity with the material. But their basic weakness, using canned sentences, makes their application suitable only as a preliminary to further work with whole sentences and paragraphs.

Thus, the system addresses the issue of students with weak entry skills by offering pre-instruction. And once on the system, students can access HELP at any time. Further, brief monitoring[3] by human teachers can further reduce problems of misinformation between the student and COMCON. But the harsh fact remains that whatever precautions are taken, whatever heuristics added to HELP and COMCON to check against student error, misinformation is bound to exist. What are its consequences?

In a carefully designed system (I am fully aware of the inexactitude of that phrase), misinformation will always arise in some typical cases. There will be some tests for sentence features bound to fail or applied incorrectly, but if the majority do hold, then the system will properly analyze the majority of input sentences. Preliminary use with students (in very limited and so statistically meaningless numbers) has suggested that this is so. Whether there are remaining problems sufficient to defeat the system is an empirical issue; for reasons given in the section on implementation, a full-scale study is not possible at this time. But there are indicators that using the student as a semantic processor does not entail great risks. Let me show why.

When HELP asks for information, it also asks the student to perform a test at the same time. A typical one requires, for instance, the placement of a determiner in front of a word to see if it is a noun. Suppose the word in question were "Running" as in "Running quickly, I caught up to Tom." One would have to wonder about a great deal more than punctuation problems for a student who would think "The running quickly" was an acceptable English utterance. By the same token, putting "The" in front of "running" in "Running is healthy" yields an awkward but recognizable English expression that few people would miss. A similar test on verbs asks the student if he could take a word and place it in, say, the past tense. Few persons, indeed, would think "Runninged" possible; their skills would be obviously so deficient that they would be unlikely members of a college-level (or even high school) English course, remedial or not.

So the risk the system must take seems to me not very great. Moreover, with the student's information, what COMCON or HELP can do is greatly expanded. A "Don't Know" from the student would not make the system any worse off than before it asked the question. A wrong answer would make a difference, but as I hope I've shown, the risk of a wrong answer is calculated and, I think, acceptable.

IMPLEMENTATION

There are times, alas, when one is all dressed up with nowhere to go. Such is the case, for the time being anyway, with COMCON, SENTCHECK, and HELP. They are all expensive programs to run since they rely on doing scans on each input sentence. Moreover, taken together they occupy seven APL workspaces (at about 70K bytes each) as well as requiring use of the APLSV file system. Since APL is an interpretive language, the combined program uses close to a half to three-quarters of a second of CPU time for each round of instruction. Given the billing policy at the present installation (the Ministry of Education of the Province of Quebec), this means that it costs more than a dollar each time a student enters his sentence and receives instruction. Since this process can take up to five minutes (depending on how much of the HELP facility is used)), there might be about ten or so sentences involved per hour. Given the realities of funding in the Province of Quebec, these costs are prohibitive, especially when they are multiplied by twenty (the number of students in a typical remedial class at Vanier College) and multiplied again by five or six (the number of remedial classes offered each semester).

Moreover, the time-sharing environment, at least the present one, is not a good one for teaching. Delays of up to ten or twenty seconds are common with one or two minute delays not unknown. Add to this much transmission of noise during peak hours because of the sheer volume of users, system crashes and assorted malfunctions, the problems of showing naive users how to live with a large, powerful, and not particularly friendly computer system, and one has a educational horror show. But these are familiar problems, I suspect, to anyone with time-sharing experience except for those on the few systems, perhaps, dedicated to CAL.

The obvious solution is, naturally, a stand-alone computer. It will be, however, a couple of years yet until a computer powerful and affordable enough to handle this program is available. It is surely coming; whether the funds to support the implementation and further research also will be forthcoming is another question. Given the cost of running the program as part of a regular instruction, there is no possibility of this in the next year. Even testing with individual students will be curtailed until such time as a device is developed--and sold at a reasonable price--to make the whole enterprise affordable.

CHAPTER VII
DISCUSSION AND CONCLUSIONS

The major intention of this thesis has been to demonstrate the feasibility of an intelligent CAL program in the teaching of writing skills. To that end, a program, COMCON, has been produced which can take as input any sentence the student writes, form an internal representation of it, and enter into a conversation about the correctness of the commas in that sentence. This gives COMCON a considerable advantage over frame-based CAL in that rather than being the generator of variety (in the form of canned sentences and canned answers), it can respond to the truly important variety, the student's. From the perspective of cybernetic theory, this capability is essential if one hopes to "regulate", that is, teach, the student. From the perspective of improving transference, COMCON's instruction goes beyond having the student identify errors in existing sentences to writing his own error-free ones. This is superior to frame-based CAL since composing error-free sentences is far closer to the writing of essays, the major activity in freshman English or remedial courses.

COMCON has three major components: a set of production rules which creates an internal representation of the student's sentence, a matrix-based model of the student, and a suite of goal-directed programs that control the conversations between student and system. I have argued throughout that an effective CAL lesson to teach writing skills must have all three. To the extent they are lacking, the program cannot accept the full range of the student's variety, have an adequate sense of the substance and structure of his knowledge, nor be able participate in meaningful conversations with him about the material. I have, therefore, placed my proposals for these three components at the center of this thesis.

To meet the demands of the variety generated by the student, I have used a set of production rules which represent knowledge about using the comma. These production rules, a group of IF-THEN statements, cause COMCON to look for certain features and their configuration in sentence input and then either take or suggest appropriate actions as a result of its search. The formalism appears well-suited to the task set for it here. It can be used to examine the entire string the student enters, rather than merely portions of it like procedure-based programming languages. Hence, it is extremely flexible and able to respond to a wide range of data. As I have shown, it can be used either to establish or confirm an hypothesis, and it is well-suited to accommodating logical implication. It has at least some of the power of the semantic net in marking relationships but makes manipulating them easier. Because it requires no complex notation such as one finds with predicate calculus, the production rule formalism is easy to document, maintain, and understand by either the original author or by others. Moreover, its modular nature allows new rules to be added quite easily, permitting extensions to the teaching program. And this applies not only to the rules themselves but to the "metarules" or filters which fine-tune the way the system examines data. So, should an instructor decide that no comma is needed after a prepositional phrase, that rule could be deleted. Or if he wished to include other sorts of punctuation, they could be added once the proper rules had been formulated.* Although not implemented on COMCON, there are a number of programs in the development stage which deal with the case of pronouns and dangling constructions which later versions will incorporate.** These changes could be readily included in the sorts of conversations COMCON produces without disrupting other elements already there. In brief, the production rule formalism is powerful,

----

* For instance, "IF the string after a semicolon is not a main clause and IF it is not part of a series of items, THEN the semicolon is wrong".

** "IF a pronoun follows a form of the verb 'to be', and IF it is in the objective case, THEN it should be changed". Or "IF the first word in a sentence is a present participle AND IF it does not refer to the subject of the clause THEN the participle dangles".

flexible, and congenial.

Like, many other researchers, I have discovered that one cannot use CAL effectively without having a great deal more information about what the student knows and the way he knows it. The matrix model of the student described offers, I think, an efficient method of his looking at the substance of knowledge ("He can see that X is a subordinating conjunction") as well as its structure ("He can see X is a subordinating conjunction but not that it begins an introductory clause"). The three-valued logic used in the matrix distinguishes whether something is right or wrong (or true or false) as well as what is undecided. One has a way of seeing, therefore, what has, could, or should be done.

The matrix-based model of the student also allows us to look at the student's writing both in terms of his most recent sentence and all the sentences he has written. This permits us to converse with him about his current work and also to decide what kinds of remediation are indicated over the long run. The model does not have the predictive powers to show what topic should be learned next, but it can say what has not been learned on the very strong evidence of the student's own sentence. One potential shortcoming is that it represents the student's actions as a subset of those an expert would perform. However, because the student model not only gives information about the student but also includes a comprehensive model of his sentence, one can argue that there is not a vast difference in seeing what the student does and what he must do. Indeed, if the model cannot account for every contingency, it can account for those most likely to occur in student writing. The considerable amount of book-keeping necessary to keep track of all this information is, I suggest, one of the most convincing arguments for using a computer to teach in the first place.

COMCON's third major component is a suite of goal-oriented programs that controls the actual conversing with the student. The basic strategy of the conversation programs is to direct the student's attention to potential errors and to suggest ways to him of correcting them. It helps him to see the analogies that should exist between the model of the properly-punctuated sentence and what he has written. To do this, however, requires COMCOM to ask the student to supply the semantic information it itself cannot. Using the student's knowledge raises a number of questions. First, a program that requires a human being to provide it with essential information does not demonstrate genuine intelligence. This is true, of course, but the issue in this application is not to provide the student with a perfectly punctuated version of his sentence (even if it were possible, what would be learned?), but to help him learn to do his own corrections. Like MYCIN, COMCON is not meant to replace the student but to augment him. I suggest on the basis of my experience with teaching grammar that engaging the student in this kind of conversation causes him first to introspect and then exteriorize his notions about the structure of his sentences. Such awareness is a key part of learning to use commas. Therefore, the program makes liberal use of "what-about" questions because they direct the learner's attention to the analogies that must exist between the structure of what he has written and the structure of the model he has been taught. In fact, "what about" questions cause the student to look for the very features which the program itself is looking for; the student punctuates his sentences using a variant of the same algorithm the program's "master performer" uses. Each new sentence becomes a new opportunity to use the algorithm in a new situation, and so accords an opportunity for transference of skills to occur.

Indeed, the student has an added advantage over the computer when making corrections in that he can readily understand semantic information with which the machine struggles. In effect, COMCON's strategy combines the computer's capacity to recognize many patterns ("abc" with a comma before it, "xyz" at the start of a sentence followed by a comma, etcetera).with the human capacity to clarify contexts. So while it is true that in many cases the computer must interact with stodent in order to be a master performer, that very interaction is valuable for the student.

A second objection might be that the student could easily feed COMCON wrong information. For example, asked if "but" joined two main clauses, the student might not know what constitutes a main clause. There are two answers to this. First, COMCON --like all instruction--requires a certain entry level, and so a screening process must be used. In the present implementation, this takes the form of pre-instruction sessions in which the necessary entry skills are taught. Eventually, however, I intend to use the branching programs I have spoken of which do succeed in teaching the simpler topics (such as "'and' is a coordinating conjunction"). Secondly, whatever form the screening process takes, it need not be absolutely perfect because COMCON contains a HELP facility that can be called upon at any time to teach what is required.

COMCON, then, provides a way not merely of correcting errors but of developing strategies for correcting errors. These range from learning discrete facts ("X is a preposition") to procedures for integrating those facts into a repertoire that recognizes and acts upon the analogies between what has been written and what should have been written. And this repertoire of skills is the one which the student must call upon when he writes his essays.

On the issue of implementation, I have said that using a stand-alone computer would be preferable to using a time-sharing environment because the latter is both expensive and troubleprone. But, in fact, I have not been able to use standalone microcomputers; the few that do exist are too expensive at the moment. The problem of cost, however, is one that should become less acute. Within a very few years, we should have available systems powerful enough to run COMCON in extremely user-friendly ways and cheap enough to buy in sufficient quantities.

Future Research

This thesis has been a feasibility study with only limited experiment with students. One hopes that, resources permitting, the next step is a medium-size try-out of about thirty students to determine which of COMCON's components needs further development. I would like to test the hypothesis that COMCON produces a significant transfer of the skills needed to write essays. Because costs are high in producing programs like COMCON, there should be evidence to support further development. However, as I have said, this kind of experiment requires resources not available at the time of this writing.

A second major area of future work is the development of new production rules and the refining of the present ones. These need not be limited to the comma but can extend over the whole of the freshman and remedial writing course material. This clearly would be useful at the practical level of writing new CAL, but would also contribute to determining the value and limits of the production rule formalism. Indeed, there remains the very much broader issue of generalizing the techniques I use here to other subjects. Along with this, one would like to see further development of algorithms and student models.

Thirdly, although I have not included the actual computer code in this thesis, I can assure the reader that it needs much improvement. COMCON's code evolved slowly, painfully, and almost entirely without the help of professional programmers. The result may work, it is true, but it is neither elegant nor efficient. Indeed, the present version is difficult to maintain, difficult to extend, and difficult to afford. Before a full-scale implementation can take place, there would have to be a thorough revision of the programs.

Finally, I realize that much of what I have said here criticizes a large portion of existing CAL. I hope it is clear I did not intend to disparage or to be hostile to the fine work so many people have done. But I sincerely believe that we have reached an important point in the development of CAL. We can no longer congratulate ourselves just because we get a program to run or because its hardware or graphics are impressive. We cannot even be satisfied with high scores on post tests. We can be pleased with our work only when we show it truly bringing people and ideas together.

Without making this sound like a clarion call to the brave and to the heroic, we must face head-on questions both difficult to answer and difficult to confront. It is all too easy--having invested so much of ourselves in our programs--to shrink from the harsh questions of whether they have any real educational value. And these questions are of more than academic interest. As computers become increasingly present in our schools, as assuredly they will, we have to be able to articulate their functions. And I mean "articulate" both in the sense of expressing ourselves in words and of understanding how computers are "jointed" with other components in educational systems. If we can't do this, we risk turning a potent technology into just another gadget that plugged into a wall and which once had such great promise. It is up to us as educational technologists to speak to the hard issues, and to that end, I hope that this thesis has contributed.

BIBLIOGRAPHY

Ashby, W. Ross (1964). An Introduction to Cybernetics, London: Methuen.

Boden, M. A. (1977). Artificial Intelligence and Natural Man. New York: Basic Books.

Brown, J.S., & Burton, R.R. (1974). SOPHIE--A Pragmatic Use of Artificial Intelligence in CAI. Proceeding of 1974 Annual Conference of ACM.

Beer, Stafford (1974). Designing Freedom. Toronto: CBC Publications.

Beer, Stafford (1975). Platform for Change. London: Wiley.

Beer, Stafford (1981). Death is Equifinal an address to the Society for General Systems Research, Toronto, 1981.

Boyd, G. (1971). The Appropriate Level of Sophistication of Computer Languages for the Writing of Tutorial Models. Aspects of Educational Technology V (1971) D. Packham et al., London: 1971

Brown, J.S., Burton, R.R., & Zdybel (1973). A model-driven question-answering system for mixed-initiative computer-assisted instruction <u>IEEE Transactions on Systems, Man, and Cybernetics</u>. Vol. SMC-3, No.3, May 1973, pp. 248-257.

Buchanan, B., Sutherland, G. & Feigenbaum, E.A (1969). Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry, in <u>Machine Intelligence 4</u>, American Elsevier, New York.

Bunderson, C. Victor and Faust, Gerald W. (1974). Programmed and Computer-Assisted Instruction, in The Psychology of Teaching Methods, 1976. Chicago: The National Society for the Study of Education.

Carbonell, J.R. (1970). AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction. <u>IEEE Transactions on Man-Machine Systems, 11(4)</u>.

Dreyfus, H.L. (1972). <u>What Computers Can't Do: A</u>

Critique of Artificial Intelligence. New York: Harper and Row.

CONDUIT (1977). 1977 CONDUIT State of the Art Reports for Selected Disciplines, Iowa City, Iowa: CONDUIT.

Crowder, N.A. (1964). On the Differences between Linear and Intrinsic Programming, in De Cecco, J.P., Educational Technology, Readings in Programmed Instruction. New York: Holt, Rinehart, & Winston.

Davis, R, Buchanan, B, & Shortliffe, E. (1977). Production Rules as a Representation for a Knowledge-Based Consultation Program, Artificial Intelligence, vol. 8.

Edwards, J., Norton S., Weiss, M., and Van Dusseldorp, R. (1975). How effective is CAI? A review of research. Educational Leadership, 33, 147-153.

Gable, A. & Page, C.V. (1980). The Use of Artificial Intelligence Techniques in Computer-Assisted Instruction: An Overview. International Journal of Man-Machine

Studies, 12, 259-282.

Gagne, Robert M. (1965). The Conditions of Learning. New York: Holt, Rinehart, & Winston.

Gagne, Robert M. and Briggs, Leslie J. (1974). Principles of Instructional Design. New York: Holt, Rinehart, & Winston.

Garrison, Karl C. and Magoon, Robert A. (1972). Educational Psychology: An Integration of Psychology and Educational Practices. Columbus, Ohio: Charles E. Merrill.

Gerlach, V.S., Reiser, R.A., and Brecke, F.H. (1975) Algorithms in Learning, Teaching, and Instructional Design. Technical Report 51201, College of Education, Arizona State University, Tempe.

Goldberg, A. (1973). Computer Assisted Instruction: the application of theorem proving to adaptive response analysis. Technical Report 203. Stanford University Institute for Mathematical Studies in the Social Sciences.

Goldstein, I (1979). The genetic graph: a representation for the evolution of procedural knowledge. International Journal of Man-Machine Studies, vol. 11, 51-87.

Hartley, J.R. (1978). Programmed Learning and Educational Technology, 15.

Hausmann, K. (1979). Instructional Computing In Higher Education. AEDS Monitor, 18 (4,5,6),32-37.

Howe, J.A.M. (1978). Artificial Intelligence and Computer-Assisted Learning: Ten Years On. Programmed Learning and Education al Technology, vol. 15, no. 20, pp. 114-125.

Judd, Charles H. (1908). The Relation of Special Training to General Intelligence. Educational_Review, 36, 36-37.

Kearsley, G.P. (1977). Some Conceptual Issues in Computer-Assisted Instruction. Journal_of_Computer-Based Instruction, vol. 4, no. 1, 8-16.

Keller, A. (1979). A_Comparison_of_CAI__and__PI__in__The Teaching__of__the__Mechanics__of__English. M.A. Thesis, Concordia University, Montreal.

Keller, A. (1980). The Design of an Adaptive CAI Program for Writing Skills, 1980__Conference_Proceedings_of_the Association__for___the___Development___of___Computer-Based Instructional_Systems, 41-44.

Kimball, R.B. (1973). Self-Optimizing computer-assisted tutoring: theory and practice. Technical_Report_No.___206 (Psychology___and___Education___Series), Institute for Mathematical Studies in the Social Sciences, Stanford

University.

Klausmeier, Herbert J. and Ripple, Richard E. (1971). Learning and Human Abilities. Third Edition. New York: Harper and Row.

Landa, L.N. (1974). Algorithmization in Learning and Instruction. Englewood Cliff, New Jersey: Educational Technology Publications.

Landa, L.N. (1976). Instructional Regulation and Control: Cybernetics, Algorithmization and Heuristics in Education. Englewood Cliffs, New Jersey: Educational Technology Publications.

Kline, Edward A. (1977). Computer-Assisted Review Exercises: English as a Second Language. In Stantz, M. and Motsinger, L. Proceedings of The Indiana University Computing Network, 4th Annual Conference on Instructional Computing Applications. Fort Wayne, Ind: Indiana University at Fort Wayne, 30-35.

Koffman, E.B. & Blount, S.E. (1975). Artificial Intelligence and automatic programming in CAI. In Lecarme, O & Lewis, R. (eds), Computers in Education, North Holland, Amsterdam.

Merrill, H. David, Schneider, Edward W., and Fletcher, Kathie A. (1980). TICCIT Englewood Cliffs, New Jersey: Educational Technology Publications.

Newell, A. & Simon, H.A.(1972). Human Problem Solving. Englewood Cliffs, N.J.: Prentice-Hall.

Norman, D.A. (1979). Analysis and Design of Intelligent Systems. In Klix, F. (ed). Human and Artificial Intelligence, Amsterdam, North Holland Publishing Company.

O'Shea, T. (1979). A Self-Improving Quadratic Tutor. International Journal of Man-Machine Studies, vol. 11, 97-124.

O'Shea, T. (1979a). Self-Improving Teaching Systems.

Basel: Birkhauser Verlag.

Pask, G. (1969). Adaptive Machines, in Davie, I and Hartley, J. Contributions to an Educational Technology. London: Butterworth.

Pask, G. (1972). Anti-Hodmanship: A Report on the State and Prospects for CAI, Richmand Surrey, System Research Ltd.

Pask, G. (1975a). The Cybernetics of Human Learning and Performance. London: Hutchinson.

Pask, G. (1975b). Conversation, Cognition and Learning. Amsterdam and New York: Elsevier.

Pask, G. (1976). Conversation Theory: Applications in Education and Epistemology. Amsterdam and New York: Elsevier.

Peplinksi, C. (1970). A Generative CAI program that teaches algebra. Technical Report No. 90, Computer

Science Dept., University of Wisconsin, Madison.

Rockart, J.F. and Morton, M.S. (1975). Computers and the Learning Process in Higher Education. New York: McGraw Hill.

Schank, R.C. (1977). Conceptual dependency and knowledge structures. Proceedings of the 5th International Joint Conference on Artificial Intelligence, pp. 988-989.

Self, J.A. (1974). Student Models in Computer-Aided Instruction. International Journal of Man-Machine Studies, vol. 6, 261-276.

Self, J.A. (1979). Student Models and Artificial Intelligence. Computers and Education, vol. 3, 309-312.
Sleeman, D.H. (1975). A problem solving monitor for a deductive reasoning task. International Journal of Man-Machine Studies, 7, 182-211.

Sleeman, D.H. and Hendley, R.J. (1979). ACE: A System

Which Analyses Complex Explanations. International Journal of Man-Machine Studies, vol. 11, 125-144.

Smallwood, R.D. (1962). A Decision Structure for Teaching Machines. Cambridge, Mass: MIT Press.

Stansfield, J.L. (1974). Programming a Dialogue Teaching Situation. PhD dissertation, School of Artificial Intelligence, University of Edinburgh.

Tait, K., Hartley, J.R., and Anderson, R.C. (1973). Feedback Procedures in Computer-Assisted Arithmetic Instruction. British Journal of Educational Psychology, 43, 2, 161-171.

Taylor, E. F. (1968). Automated Learning and its Discontents. American Journal of Physics, vol. 36, 496.

Thames, Anna Maria (1972). CAI and English: A Tentative Relationship, in Proceedings of the 1972 Conference on Computers in Undergraduate Curricula. Atlanta: The Southern Regional Education Board, 305-310.

Thorndike, E.L. (1913). *The Psychology of Learning: Educational Psychology.* New York: Teachers College Press.

Thomas, D.B. (1979). The effectiveness of computer-assisted instruction in secondary schools. *AEDS Journal*, 12(3), 103-116.

Waterman, D.A. & Hayes-Roth, F. (1978). An Overview of Pattern-Directed Inference Systems. In *Pattern-Directed Inference Systems*, Waterman, D.A. and Hayes-Roth, editors. New York: Academic Press.

Waterman, D.A. & Hayes-Roth, F (1978B). Principles of Pattern-Directed Inference Systems. In *Pattern-Directed Inference Systems*, New York: Academic Press.

Wertheimer, Max (1959). *Productive Thinking.* New York: Harper & Row.

Wexler, J.D. (1978). Information networks in generative

Computer-Assisted Instruction. IEEE Transactions on Man-Machine Systems, 11(4), December.

Winograd, Terry (1974). Five Lectures on Artificial Intelligence. AI Laboratory, Memorandum AIM no. 240, Stanford University.

Winograd, Terry (1972). Understanding Natural Language. University Press, Edinburgh.

Winston, Patrick (1977). Artificial Intelligence. Reading, Massachusetts.

APPENDIX I: A MODEL OF THE PROPERLY-PUNCTUATED SENTENCE

Recall the model I gave earlier:

(I1,) MC1(DC1),ccMC2(DC2)(,NE,)(S)

Let us examine it in more detail by reconstructing its evolution from a simple sentence. Consider the following:

S1: The man threw the ball.

Here the entire string is a single main clause (which we will call MC1). It is semantically complete; it has a subject and predicate like other clauses and it can also stand alone and make complete sense. If we add an introductory element, the result is S2:

S2: After lunch, the man threw the ball.

Notice that a comma was placed after "lunch". Although some writers consider this unnecessary since it is a brief prepositional phrase, the model insists on a comma being there. This insistence comes both from logical and pedagogical attempts to avoid confusion one finds in S3:

S3: Above the sun shone brilliantly.

The absence of the comma after "above" forces the reader to backtrack to locate the main clause. While it is true that most sentences do not cause such misunderstandings, enough do for one to insist on a comma for reasons of logical consistency and for teaching students who are not always alert to the confusions of their sentences.

Indeed, one can say that the whole issue of commas after introductory elements is a function of the word-order syntax of English in which we know what we know by what comes first. Since we normally expect the subject to precede everything else, we must have a signal of some sort when this is not the case. English is partially inflected, but not enough, for example, to allow us to switch the words in S4 and S5 with impunity:

S4: Tom hit John.

S5: John hit Tom.

The order in which we receive the words "Tom" and "John" determines the meaning of the sentence. And extending this notion, we must be alerted when the subject is delayed--whether by a dependent clause, a phrase of some sort, an interjection, or anything else that acts as an introductory element.

To return to our main clause and its introductory element, we can, of course, add a second main clause (MC2), but it cannot simply be attached with a comma as in S6:

S6: After lunch, the man threw the ball, the boy caught it.

S6 is a "comma splice" (that is, two main clauses joined only by means of a comma). It is a frequent error in student writing. The sentence needs a coordinating conjunction after the second comma to show that the two ideas expressed in the clauses are connected. The comma alone is insufficient because its primary function is to separate units, not join them. The corrected version of S6, therefore, is S7:

S7: After lunch, the man threw the ball, and the boy caught it.

And just as clearly, it would be wrong to cast the sentence as MC1MC2 (S8) or MC1ccMC2 (S9):

S8: After lunch, the man threw the ball the boy caught it.

S10: After lunch, the man threw the ball and the boy caught it.

S9 would be considered correct by some writers because the two main clauses are rather short. Indeed, there is not much reason to worry about confusion here, but the model insists on the comma for the benefit of the student writer who should not have to worry about what constitutes the vague description "rather short". And few, I suspect, would argue that the comma is outright wrong, rather than optional. Including it, then, does no harm and much good from the perspective of the student.

Our sample sentence can be expanded still further by adding dependent clauses. For instance:

S10: After lunch, the man threw the ball because it seemed like fun, and the boy caught it.

S11: After lunch, the man threw the ball, and the boy caught it when it came to him.

S12: After lunch, the man threw the ball because it seemed like fun, and the boy caught it when it came to him.

We can model these sentences this way:

S10: I,MC1DC1,ccMC2

S11: I,MC1,ccMC2DC

S12: I, MC1DC1,ccMC2DC2

The thing to note here in terms of punctuation is that there is no comma between the main clause and its following dependent clause. This is the case since the dependent clause is not a meaningful unit of its own, but one that must be attached to a main clause. A comma (whose job is to separate) would break that connection. The only exception to this is when a following dependent clause is nonessential, a concept I will discuss below.

Note also that a dependent clause preceding its main clause is treated as an introductory element; a comma, therefore, must follow it as in S13:

S13: Because it seemed like fun, the man threw the ball after lunch.

Since the subject does not come at the beginning of the sentence, there must a signal to the reader to suspend a decision about whom this sentence has for its subject. And notice that by shifting the prepositional phrase to its usual position after "ball", no comma is allowed between the main clause and the phrase that modifies it.

Two more expansions to the model are possible. First, we can add a series of items and locate it in any of several positions. For instance:

S14: After lunch, the man threw the red ball, the white ball, and the green ball, and the boy caught them.

S15: After lunch, the man the the ball, and the boy caught them, juggled them, and threw them back.

Although the series are added to MC1 and MC2 respectively, the principle remains the same: the last item, regardless whether it is a single word or a group of words, must be separated from the others by a comma and a coordinating conjunction. Hence, "the red ball, the white ball and the green ball" is wrong. Some experienced writers will disagree, but the model removes any trace of ambiguity as, for instance, in the following:

S16: The balls were red, white, blue and green.

How many balls were there? Three (one of which was two-toned blue and green)? Or four? The comma removes all doubt, and even where there is unlikely to be confusion, it helps the student writer develop consistency. Again, the final comma is never wrong, but there are times--like S16--when its absence is misleading. Therefore, the model insists on its presence.

The second expansion adds various kinds of nonessential elements to different positions in the sentence. Here are a few examples using nonessential words, phrases, and clauses:

S17: After lunch, the man threw the ball, but the boy, alas, dropped it.

S18: After lunch, the man, who had eaten too much, threw the ball to the boy, and the boy caught it.

S19: The man, after lunch, threw the ball, and the boy caught it.

The important thing to see here is that a pair of commas enclose the nonessential element. One--either before or trailing--would be wrong because it is the whole element that is in question. Whether something is nonessential or not is, however, more of a semantic question than a syntactic one. The usual test is to read the sentence without the element and see if it still makes sense. This often is insufficient since the sentence can be grammatically acceptable even without an essential element. For instance:

S20: The man who was wearing the black hat was the killer.

S21: The man, who was wearing the black hat, was the killer.

S22: The man was the killer.

S20 says "who was wearing the black hat" is essential since one cannot identify the man without this information. S21 says the information, although interesting, is not essential to identify who the killer is, and S22 shows the sentence still is grammatically correct without the clause. S21 would be right only, however, if the context in which it was written makes clear in some other way who is the killer without making the reader rely on nonessential relative clause. So the issue is very much at the discretion of the writer to decide whether or not the information is or isn't essential. In terms of the model, however, one must enclose nonessential information in a pair of commas and omit them altogether when it is essential. One comma, to repeat what I've said, is wrong, regardless where it comes.

One further point about nonessential elements must be made. Should one come at the end of the sentence, a pair of commas is incorrect: the second one must be replaced by a period. For instance:

S23: The man threw the ball, which was all right with the boy.

Here, everything to the right of the comma is nonessential according to the writer. Compare it to S24:

S24: The man threw the ball which was all right with the boy.

S24 conforms to the model's way of dealing with dependent clauses (which are treated the same as relative clauses), but the writer can override the rule about following dependent clauses by claiming the clause is nonessential. Once again, the issue is semantic, not syntactic. As long as the writer can make a reasonbable case for the information being merely interesting as opposed to essential, the comma is right.

The model also shows where the comma is not permitted although these circumstances are less readily apparent. The comma must not, as we have seen, be placed between a main clause and a following dependent clause (unless that clause is nonessential). It is also not permitted between the subject of the sentence and its verb, between the verb and its object, and between equal elements joined by a coordinating conjunction. For example:

S25: What is needed in these dire times we live in, is honesty and courage.

S26: Although he didn't think so, he received from his friends and relations, exactly what he gave them.

S27: He spent his childhood happily swimming in the pond, and cheerfully answering letters.

In all three cases, a person might pause when speaking at the places there are commas, and so think that commas should be inserted. Because the noun clause which functions as the subject in S25 is rather long, we pause when speaking; but no comma should be inserted there anymore then in "Boys, are green". Similarily, one would not put a comma in "He hit, the ball"; the length of the object or the verb does not alter the principle of not separating them. And lastly, S27 shows the mistake of separating two equal units (here, participal phrases) which have already been joined by a coordinating conjunction. Although these last three examples of where the comma is not permitted are not made explicit by the model, I include them here because they are an important part of the instructional program which has the model as its center.

Appendix II:

A Formative Evaluation of COMCON's Instructional Effectiveness

The intention of this thesis has been to demonstrate the feasibility of constructing and running COMCON. The sentences I reproduced in Chapter VI were written by students, but those students were not part of a controlled experiment or even a full-scale tryout. However, even though the central thrust of the thesis has not been empirical, one would naturally like to know how well COMCON facilitates vertical transfer of learning. To that end, a formative evaluation of COMCON's instructional effectiveness was carried out which investigated if students who used COMCON would indeed make fewer comma errors on their essays than when they used only the frame-based lessons. The degree to which the results of this study can be generalized, however, is limited because of a number of factors which I will discuss below. This being said, the results remain strongly suggestive.

Before reporting these results, however, it is important to underline the basic premises of a formative evaluation. Educational literature most often reports research whose methodology attempts to replicate that of the natural sciences. That is, the researcher generally hypothesizes that one kind of instruction is superior to another and constructs an experiment which attempts to control all variables. Should there be an observed difference in behavior after treatment, it can be attributed to one particular factor. This methodology assumes laboratory conditions such that the effects of one and only one independent variable at a time are significant. But educational problems are rarely amenable to laboratory techniques; they are simply too complex to be reduced without distorting the interelationships of their components. Research that ignores this runs the risk, to paraphrase Beer (1974), of measuring what is convenient to measure, not what is important. Many CAL systems, for example, can say (down to milliseconds) how long a student sat at a terminal; fewer can say what he was thinking of during that time. If one does try to incorporate "real-life" conditions into a controlled study, however, there is the real difficulty of proving that despite the swirl of the environment, the factor that truly affected the results can be isolated.

A formative evaluation, on the other hand, is less concerned with proving one mode of instruction superior to another than with finding out how well each works, where its problems are, and what are the fundamental reasons for its sucess or failure. In other words, one tries to describe thoroughly a set of instructional materials before extending the concern to how well it fares against others. To that end, this study does not try to control variables as one might in a laboratory experiment comparing two competing CAL programs. It does not try to say that COMCON is superior as much as point out the job COMCON does. This is not to imply that in the future COMCON should not be tested against other successful methods of effecting transfer; for the time being, however, the interest lies in COMCON itself. The current study is, in effect, a simulation that looks at what happens when students have access to COMCON. The results reflect, therefore, an absolute level of COMCON's effectiveness and efficiency. Specifically, this evaluation set out to see if students could transfer comma skills to their own essays instead of to criterion tests. Their sucess would suggest that COMCON does in fact produce vertical rather than lateral transfer. A second--and equally compelling issue--is what needs to be done to improve COMCON as it now stands. Therefore, three kinds of information were sought:

1) scores that reflected the students' ability to transfer comma skills to their essays;

2) anecdotal comments from students on what they as users saw as the strengths and weaknesses of COMCON;

and 3) observations by the designer to supplement and extend student observations.

Method

The sample consisted of ten Vanier College students enrolled in Effective Writing, a remedial composition course. Students were placed in Effective Writing on the basis of two factors: 1) their scores on a standardized test of writing skills which showed that they were in the lowest twenty percent of all entering first-year students; or 2) their work during the first semester which had been sufficiently poor to warrant remedial help. From a pool of about sixty-five such students, only ten were available for the section in which COMCON could be used.

Data consisted of scores from essays written after using CPINS (the frame-based CAL I described in Chapter II) and after each of two sessions with COMCON. The scores (reported in Table I) are not those given for overall performance on the essays but only for mastery of comma skills, the subject matter taught by both CAL programs. The scores were derived from the concept of a "comma event" which was defined as:

1) an occasion where a comma was called for and inserted;

2) an occasion where a comma was called for but omitted;

3) an occasion where a comma was not called for but inserted.

The second and third instances are, of course, errors. The scores, therefore, represent the percentage of times when a comma was called for and used correctly. The significance of these percentages is that they reflect the ability of the student to take the set of discrete rules about commas he had been taught and transfer them to his own essay rather than to a criterion test in which he would identify errors in prepared sentences. The scores, then, indicate whether or not vertical (rather than horizontal) tranfer is taking place.

Students were required to write essays each week.

Typically, they were on topics such as "The Worst Movie I Ever Saw" or "A Person I Know" which required no special knowledge or research. As part of the course, students completed the frame-based lessons on the comma (generally over a four-week period) and submitted a class essay. After a brief orientation in the use of COMCON, they then inputted sentences from previous essays during a thirty minute session. Again, they went home to write and submit an essay. A second COMCOM session followed and another essay was submitted.

Each essay was first read looking for all three comma events that were described above. The number of such events was noted and the essays were reread, this time recording the number of correct or incorrect commas. This score, expressed as a percentage of right answers, makes up the raw data. The essays were then read a again, this time looking at the full range of the essays' strengths and weaknesses--grammar, diction, logical argument, style, and so forth. The mark given here was the mark the students saw and was recorded as part of the semester's work. Generally, the reader commented on errors in the usual way of composition teachers. In the case of commas, however, he pointed out only that a mistake existed but not which kind. This was done to ensure that when the sentence was later used for COMCON, no special prompting had taken place that would confound the results. The study's data, then, constituted a relatively small part of of the grading process, the part directly taught, however, by COMCON.

Results

The results showing the correct use of commas on the essays submitted after each treatment are given in Table I. A one-way analysis of variance was performed (Table II) which yielded an F-ratio significant at a .001 level. An a posteriori test using Tukey's HSD ("honestly signicant difference") was carried out which showed significant differences at the .05 level for comparisions between the CPINS and the first COMCON scores and for the comparison between the two COMCON sessions. There was a significant difference at the .001 level for the comparsion between the CPINS score and the last COMCON essay.

TABLE I:

PERCENTAGE OF CORRECT COMMA EVENTS

	CPINS	COMCON 1	COMCON 2
	41	50	59
	50	63	71
	44	47	65
	52	58	65
	33	47	47
	53	65	76
	42	57	73
	47	59	74
	58	67	81
	41	47	61
MEAN	46.1	56.0	67.2
STANDARD DEVIATIONS	7.34	7.77	9.92

TABLE II: ANALYSIS OF VARIANCE

Sources of Variance	Sum of Squares	Degrees of Freedom	Mean Squares	F
Between Groups	2228.866	2	1114.433	*15.71674
Within Groups	1914.500	27	70.907	
Total	4143.366	29		

*P >.001

Discussion

The data indicate that the use of COMCON did produce signifcant differences in the ability to use commas on essays. The higher confidence level obtained for the comparsion between the CPINS essay and the second COMCOM one bears out what appears intuitively so: continued practice with COMCOM produces better performance. This suggests how COMCON is to be used. Once the basic skills are in place, COMCON can become a kind of practice lesson, rather than a piece of courseware essentially designed only for one-time use. As students encounter difficulties--say on returned essays--they could enter the problem sentences, and COMCON would take them through the correction process. COMCON sessions would be as frequent only as student need demanded. Further, because COMCON takes the student's sentence as input, there would be no question of having to determine the appropriate level of remediation.

A second a-posteriori procedure (in addition to Tukey's HSD) was a T-test comparison between scores on an essay done before students used CPINS and the one immediately after (which is reported above). No significant difference was found between them. This bears out the initial concern which led to the construction of COMCON: frame-based CAL alone does not produce vertical transfer. Since each CPINS lesson contained a criterion test and students had to show mastery of each one before proceeding, it is clear that whatever skills CPINS teaches, they are not those which can be easily transferred to essay writing. The additional use of

COMCON, however, does appear to produce results. There are, however, some caveats about these data.

Although the results indicate that students using COMCON after CPINS are able to transfer their skills to their own essays, the results remain merely suggestive, not definitive. Because of the obvious difficulties in using an ongoing class as part of research, the tryout was not, to repeat, intended in any way as a fully controlled experiment. What was being evaluated in a formative way was the instructional effectiveness of COMCON. Comparisons with other strategies are not only difficult to perform, but, in this case, they would be inappropriate since COMCON is not so much an alternative instructional method as an adjunctive one. That is, its value is for students who, having mastered certain fundamental concepts (such as recognizing an introductory element or a coordinating conjunction in a given sentence), must have practice in applying these concepts to their own writing. To claim COMCON superior, then, is to miss the point: it functions to produce a vertical transfer once basic skills are already in place.

However, even determining the effectivess of COMCON on its own terms is difficult. First, given the numbers of students available and the amount of money that could be allocated to running COMCON, the size of the tryout population had to be restricted to ten students. Naturally, more students would allow one to feel more confident about the the results obtained. However, as I have pointed out in the final chapter, standalone computers would do away with the problems of paying for processing time. Not only could one then afford more students, of course, but each student could be given virtually unlimited access. Money, then, is an issue more in the present context than in the future where COMCON would be run on such machines. A second confounding factor was that the CAL (in either form) was not the only instruction the students received. In the normal course of teaching, they heard a series of mini-lectures, took part in discussion, did exercises, sat for weekly quizzes, and--perhaps most importantly--wrote several essays. It is impossible to say that any one of these activities did not also play a significant part in the improvement of the scores. Third, one cannot discount the fact that any teaching at all would produce results with students whose previous formal instruction in commas had been poor (or even nonexistent).

Yet another confounding factor was a possible Hawthorne effect. Students knew that COMCON was still in the development stages, something which, judging by some informal comments, interested them. The breakdowns of the computer necessitated a monitor to be with them during the COMCON sessions, not to advise on commas but to troubleshoot system failures. One cannot accurately say how much this may have induced students to be particularly careful about writing their essays. It would, of course, be interesting to see essays written after the course has ended.

In summary, then, the tryout was somewhat compromised by the small size of the sample, the unmeasurable influences of other instruction, and by a possible Hawthorne effect.

On an informal and anecdotal level, the brief tryout supplied, as one would expect, some very practical advice on improving COMCON. Some of this advice was not necessary: it was no surprise that the system broke down far too often. The delay between the student's input and the system's response typically was about three to five seconds, with waits up to thirty seconds and even a minute not uncommon. This is obviously unacceptable. Moreover, not a single session went by without the system failing altogether for no apparent reason. Part of this may be due to the large amount of processing COMCON requires and its attendant yet still undiscovered bugs, but a good measure of it was due to the vagaries of an overburdened time-sharing computer system. This would have been quite bad enough without the terribly high costs (about six to seven dollars a session) for just running the program.

Student comments generally were quite favourable; I reproduce a representative sample of them here which were collected after the first COMCON session:

"It makes you see your errors . . .
here you write your own sentences and that
helps you understand better . . ."

"Some of the questions are confusing . . . what's
a string?"

"The delays really bother me . . ."

"It's helpful because it reminds me
of the steps needed. . ."

"It tells me 'why' . . . I can't do that with CPINS . . .

it's better than CPINS because you have to think twice.".

"You have to know your theory, otherwise it's no use."

"It's good, you see your mistakes . . . but
parts of the instruction is confusing . . . compared
to CPINS, we get to see what we write, it comes
out of you . . .
The same stuff as as CPINS but you get more into it."

Other than criticism about the computer's failures, most students focussed on the sometimes poor (or at least ambiguous) instructions. The use of the word "string", chosen so that the function in which it appeared could be used in many contexts, was not clear enough for most students. Making that function context-sensitive, however, is a technical problem that requires more room in each workspace.

Perhaps the single biggest design feature the tryout revealed was the need for students to have a very strong hold on entry skills. Although COMCON includes a HELP facility, students often did not know that they in fact needed to use it. So if a student thought a string was a main clause when it wasn't, COMCON failed. The HELP facility does include a way of determining if a string is or is not a main clause, but, once again, putting such aids on the main program where they will be more readily used presents a technical problem of space.

One part of COMCON's design that was not properly examined by the tryout was the use of the student model over time. Recall that COMCON's matrix model of the student not only records information about the current sentence (which worked as designed) but also about previous ones. These earlier efforts are supposed to guide COMCON as it presents advice to the student about where he is weak and what he might do next. But the relatively few sentences that were entered each session were not enough to get a real sense of the student's development. Nor was it possible to store records from session to session, although that again is a technical issue. In any event, what had been a central feature of COMCON's design simply did not come into play. Future work clearly should address itself to that.

It was particularly satisfying to hear comments that COMCOM "made me think about what I was doing". A fairly usual comment was that putting in one's own sentence was a better way of learning since it was closer to what one could or couldn't do. This is, in fact, the major claim I would make for COMCON: because it responds to the student's variety, it is more like a human teacher in finding out where a student need work. In this way, it is unlike branching CAL. CPINS, for instance, was a game students reported that they could beat just by seeing what it was teaching at that moment and answering its questions quite mechanically. One weaker student preferred CPINS, however, because she was not sure at all of the entry skills. This makes it clear that unless the student is in control of the basics --as with an algorithm-- the whole procedure can be severely compromised. How those basics are taught, however, is not an issue that directly concerns COMCON. As long as entry skills are taught sufficently well to allow the use of COMCON, they can come from any effective and efficient teacher, human or otherwise.

In summary, then, the first tryout bears out the major theoretical predictions about a system such as COMCON. Because it has more of the "requisite variety" to respond to the student's variety, it is far closer to human teaching than frame-based CAL. Having the student's own sentence as the basis for the "conversation", moreover, means that it is the student's real level of proficiency that is being addressed, not simply a guess about it. The model of the current sentence makes this possible. Most importantly, the very composing of sentences is far closer to the skills involved in writing essays. This means that the student truly receives practice in what he is trying to master. The limited generalizability of the tryout notwithstanding, there is empirical evidence to bear this out.

There are a number of problems, of course, still to be solved. One must get rid of the technical failures in running COMCON; the breakdowns and delays of the system will otherwise defeat one's best efforts. The problem remains one of securing and programming the right hardware. In terms of instructional design, some of the "what about" questions need clarification. The current version contains a number of instructions that confuse more than they teach. The lack of "theory", as one student described the absence of basic materials (such as "a main clause is . . ."), is both a technical problem of space and a pedagogical one of determining the precise roles for the two CAL modes. This remains for COMCON's future development. But overall, one can only be encouraged by the results of this tryout.