ED 163 974 IR 006 647

AUTHOR Davis, Mark Edward; Pettit, Teri

TITLE Using VOCAL: A Guide for Authors. Technical Report No. 296.

INSTITUTION Stanford Univ., Calif. Inst. for Mathematical Studies in Social Science.

SPONS AGENCY National Science Foundation, Washington, D.C.

PUB DATE 25 Aug 78

GRANT EPP-74-15016-A01

NOTE 32p.; For related documents, see IR 006 644-647

EDRS PRICE MF-$0.83 HC-$2.06 Plus Postage.

DESCRIPTORS Auditory Perception; *Computer Assisted Instruction; *Curriculum Development; Display Systems; *Guidelines; Instructional Design; Objective Tests; Programers; *Programing Languages

IDENTIFIERS *VOCAL

ABSTRACT

Factors which affect the overall presentation of computer assisted instruction in the Voice Oriented Curriculum Author Language (VOCAL), and general guidelines for effective use of VOCAL in curriculum development are presented. Four major sections describe course design and review, spoken strings, display features, and six types of exercise questions. (CMV)

ED163974

USING VOCAL: A GUIDE FOR AUTHORS

by

Mark Edward Davis and Teri Pettit

TECHNICAL REPORT NO. 296

August 25, 1978

PSYCHOLOGY AND EDUCATION SERIES

INSTITUTE FOR MATHEMATICAL STUDIES IN THE SOCIAL SCIENCES

STANFORD UNIVERSITY

STANFORD, CALIFORNIA 94305

IR006647

Acknowledgments

The authors wish to thank Patrick Suppes for his able direction of this project. We are also indebted to Robert Laddaga, whose perceptive editorial comments and criticism has been of great value in clarifying many points which might otherwise have been vague. Our conversations with him, as well as those with VOCAL users Mike Hinckley, Arvin Levine, Ingrid and Sten Lindstrom, and Wilfred Sieg, have suggested most of the topics which are covered in this manual. The work described here was supported under NSF Grant EPP-74-15016 A01.

Table of Contents

1 INTRODUCTION

In this manual, we attempt to help VOCAL authors to see what factors will affect their overall presentation, and give some general guidelines for effective use of VOCAL. It is written by a group of people who have used VOCAL over the past year to write courses in logic, proof theory, set theory, probability, and social decision theory. Our writing has improved gradually as we have gained experience in using the language. We hope to provide new VOCAL authors with some vicarious experience so that their progress may be a little more rapid.

1.1 Course Design

VOCAL is a new language, and as of this writing has only been used to present mathematical or meta-mathematical subject matter, in conjunction with a sophisticated automatic proof checker. This type of course benefits greatly from an interactive, computer assisted teaching environment. However, the recent rise in popularity of "programmed instruction" workbooks for college-level introductory courses in psychology, biology, and other subjects suggests that VOCAL could also be an effective tool for non-mathematical courses. Short of experience with other computer-assisted courses, familiarity with the workbook type of programmed instruction is perhaps the best background for designing courses to be implemented in VOCAL.

Frequent solicitation of student responses, and a gradual increase in the complexity of skills required in those responses, are the hallmarks of all programmed instruction. But VOCAL provides much more flexibility, even when restricted to TTY (teletype) mode, than is possible either in a workbook or with older CAI author languages. This is especially true in the analysis of wrong responses, as is made apparent in Section 5, on Questions. While most programmed instruction has ignored the entire area of diagnosing errors, and traditional person-to-person teaching can put off dealing with specific problems until they arise, VOCAL writing lets one take the midroad of predicting the types of errors which will be made, and writing separate responses for each anticipated error. In addition, the combination of VOCAL's audio and display facilities enable one to go step by step through examples of complicated procedures in a very natural way. Although it is possible to design a course, in the sense of outlining the topics to be covered, without taking into account these features of VOCAL, your course will have a tighter structure and a more unified style if you make some decisions about how much error diagnosis will be done, and what kind of examples will be given, while still in the early stages of course design.

An obvious but important point is that VOCAL is only a tool, a very sophisticated teaching aid. No amount of proficiency in the language can bring direction and structure into a poorly designed course or make sense of a muddled explanation. A prospective VOCAL author should have (1) a very clear idea of how the course he is writing is organized, (2) a skill (or talent) for explaining potentially confusing material in a way which students will understand, and (3) a knowledge of how to make the best use of the interactive capabilities which are provided by VOCAL. This manual can only provide help in the latter category.

1.2 Reviewing your Work

It is difficult enough, in many cases, to be able to effectively and objectively review the faults in one's own writings. With the added dimensions that one is given in using the VOCAL language, this is usually even more difficult. The beginning VOCAL author is often oblivious to various problems in his organization and use of VOCAL which make the presentation of his work irritating, inefficient, or unaesthetic. Certain approaches he may take will turn out to be time-consuming and wasteful for him as well. One partial solution for the author is to always have his work reviewed by a second party. A further answer is supplied by the present manual, which is based upon observation of the many sorts of errors made by beginning authors.
WARNING: Unfinished TVEDIT

2. SPOKEN STRINGS

It is a hazardous VOCAL practice to assume that the prosody program or the person recording speech segments for Long Sounds mode will be able to guess how you want abbreviations, acronyms, or special symbols to be pronounced. Consequently, all such 'words' should be written out explicitly in the form you intend them to be spoken. In particular, anything which is to be spelled aloud should be written with spaces between the letters. Single quotes should be put around the resulting phrases to indicate to the prosody program that they should be parsed as nouns. (Quotes will also warn a human reader that something unusual is coming up.) Here are some examples:

BAD FORM	GOOD FORM
.7	\`point 7´
Rule AA	\`Rule A A´
X*	\`X star´
#2	\`number 2´ or \`sharp 2´
Dr. Jones	Doctor Jones

One should also avoid including in speech segments abbreviations such as "i.e." or "e.g.", which are meant for written text. Use "that is" and "for example" instead.

Numbers which are longer than one digit should also generally be written out as words (or as combinations of words and single digits). No matter what digits-to-words convention is used by the program, it is bound to be inappropriate in some contexts. For example, the number "206" should probably be read as "two hundred and six" if it occurred in the context: "Last year, 206 students enrolled in this course", and as "two oh six" in the context: "In Lesson 206 we introduced the Definition of Subset." The task of making such distinctions properly belongs to the lesson author, rather than the program or the person recording for Long Sounds.(1)

If Long Sounds mode is to be used, speech segments should be broken whenever possible at natural pauses or constituent boundaries. Since each string is recorded separately, this practice will make the recording easier for a human reader, and the speech will sound more natural when the resulting Long Sounds are output. See the sample code in Section for examples of properly broken speech segments.

(1)The way in which multiple-digit numbers are pronounced in Prosody and Spell Mode has been changed several times. Currently, "206" would be read as "two zero six".

3 DISPLAY FEATURES

3.1 The Screen

The Datamedia screen holds 24 lines. When a student is in a lesson, the first line will contain the lesson header. In addition, a "Mark Line" is automatically placed between the scroll region and the blackboard (i.e., template controlled) region. The Mark line is blank until a Q or M opcode fills it with a row of widely spaced dots. This leaves 22 lines to be allocated to the scroll and blackboard regions; three of these should always be reserved for the scroll region since prompts such as the one below take a minimum of three lines:

[Type ESC to continue, ^A to repeat.]

*

So the maximum number of lines which should go in a template is 19. This limit can be exceeded, but the prompts will disappear behind the divider line before the student can see them (if all 22 lines are included in the template, even the * will not appear). A verbal prompt would therefore have to be included at the end of each hold. This practice would probably be very confusing to the student, and should be avoided if at all possible.

Although the source file may be written with the maximum number of characters per line permissible on your system, the coding for VOCAL display manipulation limits the width of the students' screen to 72 characters. Positioning the TEMCHAR's ('%') used in the display region on the seventy-third space in the line will help to remind the author that his display space is limited to that.

3.2 Typing in the Scroll Region

In T and TEM opcodes, the space immediately following the first quotation mark corresponds to the first column of the screen. Since it is impossible to put this space on the first column of the source file, care must be taken to avoid a "shifted over" effect. For example, the author of the following segment may have wanted the first line to be longer:

CODE:

```
    (E 4) (T "This is a multiline typed string which will appear on the
screen as having a gap at the end of the first line, since that line
will start at the left margin.")
```

OUTPUT:

```
This is a multiline typed string which will appear on the
screen as having a gap at the end of the first line, since that line
will start at the left margin.
```

One way to avoid this problem is to leave a blank line when you want to type a multi-line message in the scroll region. Thus:

```
    (E 4) (T "
This is a multi-line typed string which will line up properly since
the first line is left blank. It looks neater in the source file
than making the first line extend farther than the ones below it.")
```

In fact, you may want to make a practice of putting a blank line even above single line scroll messages, to better separate them from lines the student has typed. In some situations, such as typed hints in a derivation, you will want to avoid including these extra lines, since they would make the student's work scroll away sooner. In these cases, we recommend the following format:

```
    HINT   (T
"First derive the formula: <formula> using an indirect proof.")
```

If more than one line is to be typed with this format, remember to mentally shift the first line one space to the left.

3.3 The Format of Template Strings

The TEM opcode is also sensitive to the way in which the quotation marks are placed. We recommend using the following format:

```
(TEM  "
                    < 1 to 18 lines of text >                    %
"
```

This format will give a blank line between the header and the actual text on the screen. If the text is to be doublespaced, then two blank lines should be used to separate header and text. Situations where no blank line is wanted will be rare. A possible exception might be where a template will be used to contain a line or two of auxiliary text inside a derive class opcode, since in derivations it is important to leave as much scroll space as possible for the student to work in. In these cases, the same format which was recommended for the T opcode should be used. Again, it is important to remember that the top line should be shifted over to the margin. This is especially true when using subarea markers, since they must be lined up with the above line. For example, if we have the following format:

CODE:

```
(TEM  "This line, `|´, is to correspond to the one below           %_
                    A
    This line, `|´, is t  correspond to the one above             %1_
                    B
"
     (T 1) (B A B))
```

then the template that will be printed out is:

OUTPUT:

```
This line, `|´, is to correspond to the one below
    This line, `|´, is to correspond to the one above
```

Notice that the quotation mark at the end of the template string should be flush with the left margin. If any spaces occur before the mark on a new line, they will cause a blank line to be printed. If the last line of the template string is a line of area markers, it is also possible to put the second quotation mark at the very end of this last line. It cannot be placed at the end of a line of text which is to be displayed, since any character which occurs after the temchar marker on the same line is ignored.

These factors are not really any problem for the second string of a TEM2, since that material will not be appearing on the screen as it is. One note though; two quotation marks printed together ("") is VOCAL's way of putting a quote mark within a string, so the following format will not work:

```
(TEM2 "
                        <template string text>
""
                        <overtype string text>
"
```

3.4 Mnemonic Area Names

Generally, when you are writing the display actions for a template, the template string itself will no longer be visible on the screen, and it can be annoying to have to jump back and forth to check the names of areas you want to brighten or overtype. A careful selection of your area names can help alleviate these difficulties by making the area names easier to remember when it comes time to use them.

The best way to name full-line or multi-line areas is to number them from 1 towards 9 on the first template string, and try to reserve 2-digit numbers for areas in the second (overtype) string. The numeral 0 can also be used as an area marker, if you need 10 names for the first string. Lines which will never be called by number, but only by their subarea names, can be given very large numbers so that the small numbers will remain for naming areas which will be called by line.

The reason for reserving 2-digit numbers for the overtype string is that they can be used systematically to indicate which lines will overtype which other lines, and in which order. One good way to do this is to overtype area n first with area n1 , then with area n2 , and so on. This method feels the most natural, since each overtype sequence is in ordinary counting order. However, for templates with 10 or more areas in the first string to be typed by line number, it has several sharp disadvantages as compared to the alternate method of overtyping area n first with area 2n, then with area 3n, and so on. The first is that if you use 0 as an area name, it is impossible to overtype it with an area named, say, "02", since that name will be counted the same as "2", while it is quite possible to overtype 0 with 20. The second disadvantage (for many-area templates) of the first method is that it usually uses some numbers in the 80's and 90's for overtype areas, thus making it hard to choose a set of "very large numbers" to be used for lines which will not be called by number. (Under the first method, numbers like 19, 29, and 39 can probably be used for this purpose, but that's not as convenient or as visible as just using the 90's.) And finally, the second method enables you to use 10, 11, 12, and so forth in the first template string, while the first method necessitates jumping from 9 to the 19-29-39 sequence, since the early 'teens are reserved for overtyping area 1. Which method you choose will probably depend on how common it is for your templates to involve a lot of areas, but you should definitely choose one and stick to it consistently.

For templates with a lot of subareas, the best method is to name them from A towards Z in the order they will be referenced. One good mnemonic for overtyping subareas is to use lower case letters as area markers in the second string, and, for example, (OB D d). If there are just a few subareas in the template, an alternate way of naming them is with some letter suggested by the text in the area. For example, if at some points in the exercise you will be brightening the words "valid" and "invalid", naming the area containing "valid" "V", and the area containing "invalid" "I", will make the names very easy to remember.

3.5 Positioning the Text

Generally, even if the displayed text is fairly small, the author will want to use most of the screen in the blackboard region rather than the scroll region, by double spacing the text and surrounding it with extra blank lines. This will usually give a less cluttered appearance to the screen, as well as clarify what is expected of the student by covering up the old prompts and student responses. If blank lines are to be added to increase the size of the blackboard region, most of them should be put below rather than above the text, since the top of the screen is closer to eye level. In TEMs which contain only one student input opcode (this includes HOLDs), these additional bottom lines are unnecessary since there will be no "old" scroll messages to cover up. The best size for the scroll region is that which will allow only the current "interaction" messages to appear. Of course, in a derive-class opcode, the interaction will be quite lengthy, so the author should make every effort to have templates within such opcodes be as short as possible.

3.6 Erasures and TEM2's

Note that at this point, overtyping a multiline area will be done line by line. That is, the first old line will be replaced by a new one, then the second old line by the second new one, and so on. This results in a rather messy "flashing" effect. It is better to either erase the old area, then overtype the old with the new, or to overerase the old with the new, then type the new area.

Whenever the entire screen is to be blanked within an exercise, it is better to use a new template than to erase and overtype with new material. It is far easier to work with a new template than to try to deal with the growing complexities of a double template, a TEM2. Moreover, when a student repeats a section of the exercise (with ^A), the display actions from the beginning of the TEM up to the time of the HOLD are repeated so as to have the same initial display. Starting with a new TEM results in less of this sort of screen-retyping. There are only two times where it seems warranted to do a complete erasure. One is when the author wishes to keep the material in the scroll region on the screen (say, in a derive-class opcode), and the second is when the erasure is desired inside a HOLD, for then it is impossible to get exactly the same effect with two distinct TEM's. These conditions, however, are fairly rare, and can usually be dealt with in another way almost as easily.

Even when a complete erasure is not wanted, it may still be better to start a new template and retype the unchanged lines rather than use a double template. The factors that would tend to keep an author from beginning a new template are the number of lines that must stay on the screen, and the amount of discontinuity introduced into the flow of information by having these lines erased and then retyped.

For example, suppose an exercise contains a question which is displayed with one line directing the student to select all the expressions satisfying some property, followed by a long list of choices. If this is to be followed by a second question which asks the student to select from the <u>same</u> list the expressions satisfying a different property, then it would probably work better to use a TEM2, and simply overtype the first direction with the second direction, leaving the list on the screen. This will move quicker, and since it will be obvious that the list was not retyped, it will also free the student from having to read the 'new' list to discover that it is the same as the 'old' list. However, if instead, the second question were to use the same <u>directions</u>, but a different list of <u>choices</u>, then the number of lines which change would be sufficient to warrant a new TEM, and probably a new exercise as well.

4 QUESTIONS

Questions in VOCAL exercises will generally fall under one of four categories: multiple-choice, true-false, translation, or short-answer. Effective use of each of these types of questions, as well as the CQ opcode, is discussed below.

4.1 Multiple-Choice Questions

This will probably be the type of question you will use most often, since the answer analysis available in VOCAL is most suited to situations where all the possible student responses can be easily itemized. A question should be typed on the display followed by a list of lettered or numbered choices. Some questions will contain only one correct choice; in others, the correct answer will be a list of several choices. Here is a list of the tags and answer analysis opcodes most useful for a multiple-choice question with several correct choices.

```
INIT   ((S (T 1)
          "Which of these .........?")
         (T "Type the letter(s) indicating your response."))
R       <number of reasonable wrong answers by single student>
A       (ALLANS <all correct choices>)
CA      <action>
WAS    ((ALLANS <subset of correct choices>) <action>
        (ANYANS <incorrect choices>) <action>
        (MATCHANY) <action>)
FAIL    <action>
```

Several things should be noted about this list. First, decide upon a 'standard' string to use as the <promptaction> in all multiple choice questions, and always type it in the scroll region. The student will get used to seeing this prompt and know that whenever it appears, the computer is waiting for a response to the question on the screen. Widely varying or spoken prompts may occasionally be confused by a student with part of the <againaction>, with the result that he will sit waiting for further explanation or prompting.

4.1.1 Number of Repetitions Allowed

The repetitions ("R") should usually be chosen as the maximum number of wrong responses which might reasonably be given in sequence by a student trying to figure out the correct answer. This way most students will eventually get the right answer without being told, which is of course better for morale.

It should be clarified what we mean here by a "reasonable" sequence of wrong responses. For example, suppose there were five letters to choose from, and the wrong answer analysis only informed the student whether a response contained a wrong choice, or whether it was incomplete. Then the student could always use the information from the analyses to figure out the correct answer in five tries, regardless of the question content. But this does not mean that R should be 5, because it is demoralizing for a student to be reduced to finding the answer by means of such a process of elimination. A "reasonable" sequence of answers is one which takes into account the content of the choices, and involves at each step the use of another major concept of the lesson. Far from being demoralizing, this kind of sequence involves a generally gratifying process of using the knowledge one has acquired, of determining which of the concepts presented are relevant to the question.

For example, if one had just presented the rules for omitting parentheses from formulas, a multiple choice question might be given in which each choice consisted of a pair of formulas — the left one fully parenthized and the right one with some parentheses omitted. The student could then be asked to indicate which of the pairs omitted all the unnecessary parentheses and no others. There are probably 3 major concepts which can be identified in this situation: (1) retain all parentheses which cannot be replaced by connective precedence conventions, (2) retain all parentheses which cannot be replaced by right-to-left conventions, and (3) eliminate all parentheses which can be replaced by either of these conventions. One can assume that a student starts out with at least one of these concepts, and should set R = 3 to give him a chance to acquire the others. (Assuming that the choices permit each type of mistake to be made.)

There are situations when R should even be smaller than the number of important concepts involved. Occasionally, you will want to point out something about the particular choices which the student omitted, and in this case repetitions would be useless since the correct answer would have been revealed in the first response analysis anyway. And when questions are included before all the information necessary to answer them with certainty has been presented (in a "what do you think it will turn out to be?" context), repetitions are uncalled for. Most questions, anyway, will probably be simple enough that they only involve one important concept (some will involve no major concepts, and be included just to allow some interaction during a "dry spell"), so for the majority of multiple-choice questions, R should simply be left at its default value of 1.

On the other hand, some authors have suggested that on major questions, R should be somewhat greater than the number of "reasonable" wrong responses, to discourage random guessing by those students who don't want to take time to think about a question. Most of us prefer to stress cooperation rather than coercion, though, presenting questions to enable students to get feedback on their understanding of the text, rather than to force them to think about it. In short, the number of repetitions you specify may be more a matter of your educational philosophy than it is of effective VOCAL usage.

4.1.2 Wrong Answer Analysis

The wrong answer analyses will generally include two types: a sequence of ALLANS's to cover responses which omit one or more correct choice but contain no incorrect choices, and a sequence of ANYANS's to cover responses which contain an incorrect choice. Note that when only one string argument appears, ANS has the same effect as ALLANS. Since the former involves slightly less computation, it is preferable in these cases. One little ``trick'' should be mentioned in this context — a sequence of ANS's which each contain one argument, and cause the same action, can be combined into one multi-argument ANS as below:

```
(COMMENT "The following list of analyses is inefficient.")
(ANS "A")
(S "You got one of the correct answers, but there are more.")
(ANS "C")
(S "You got one of the correct answers, but there are more.")
(ANS "D")
(S "You got one of the correct answers, but there are more.")

(COMMENT "This analysis is equivalent but shorter.")
(ANS "A" "C" "D")
(S "You got one of the correct answers, but there are more.")
```

Remember that whenever the WAS tag is used, the last response analysis opcode should <u>always</u> be MATCHANY. The random default reply to a incorrect answer is generated only when <u>no</u> WA, WAS, or WAL tag is present, and in fact, if the student types something which matches neither the correct answer nor one of the listed wrong answers, this is treated as if he had typed only an ESC. A new `*' line appears, but the <promptaction> is not repeated and it is not counted as an attempt at the question. So using a Wrong Answer Select without a MATCHANY may cause a student who is unfamiliar with the proper format for responses to get stuck inside a Question. Thus, whenever the preceding sequence of opcodes is sufficient to cover every possible subset of choices on the list, it will be helpful to the student to make the MATCHANY action be a reminder to separate his choices by commas or spaces.

Keep in mind that the FAIL action will be executed <u>instead of</u>, rather than in addition to, the appropriate wrong answer analysis on the final attempt. So it should notify the student that his response is wrong, as well as inform him of the correct answer.

At the end of Section 4.6 is a lengthy example of the use of CQ which includes two multiple choice questions.

4.2 <u>True-False Questions</u>

True-false and yes-no questions are so straightforward that there is very little to say about them. Use the A tag, with either (AFFIRM) or (NEGATE). Often on true-false questions this tag will be sufficient for the answer analysis; you can let the system generate a default reply. On yes-no questions, however, such a practice might cause confusion. At one time the list of strings from which a default reply was randomly chosen included "yes" for correct answers and "no" for incorrect answers. If, for example, a student correctly typed "no" and was told "yes", he might think he was being corrected, and that the answer was really "yes". We had already written quite a few lessons by the time this problem was noticed, and rather than rewrite them, the offending strings were removed from the list. It may be the case at your institution, though, that they have been re-included. After all, for any other type of question, "No" is a good way of telling someone that an answer is wrong. There is a trade off between having to avoid the default feature on yes-no questions, and being able to expand the variability available in these default replies.

Notice that no matter what the list of default replies contains, conceivably there may be cases where you will want to avoid using it. For example, confusion could also result from using this feature in a question where the student had to choose between typing "LEFT" or

"RIGHT". This is just one aspect of a more general property of computer assisted instruction, namely, that a lot is going to be happening when the student sits down at a terminal which you as a VOCAL author do not explicitly control. When writing lessons, it is important to keep in mind the actions which will be occurring `automatically´, since they will have ramifications for the material which you specify in the source file.

4.3 Short-answer Questions

Under this category we include all questions, other than translations, in which the response is not known to be limited to a subset of some small set of possible choices. Perhaps `free-answer´ would be a better description. Many questions which seem at first to be short-answer questions are in fact just multiple-choice questions in disguise, and should be treated as such in the answer analysis. Some examples of this type of question are:

```
INIT  ((S (T 1) "Which line in this invalid derivation"
           "violates a restriction on the quantifier rules?")
      (T "Type the number of the line which introduces an error."))
```

or:

```
INIT  ((S "Which of the axioms for commutative groups"
          "implies that nine plus zero is equal to nine?")
      (T "Type the (short) name of one of the five axioms."))
```

Another type of `pseudo´ short-answer question which is occasionally found is the `copy me´ request, used to give the student his first experience with typing some new symbol. Several attempts should usually be allowed. An example follows.

```
[Q
 INIT  ((S "Type the symbolic version of the quantifier:"
           "`For all A´.")
       (T "Type:  (A A)"))
```

```
   A       (EXACTANS "(A A)")
   R       2
   WAS     ((ANS "(a a)")
           ((COMMENT "This will also match `(A a)' or `(a A)'.")
            (S "The $2 A for the quantifier and the $2 A for the variable"
               "must both be $2 capital letters."
               (W 500)
               "Try again."))
           (MATCHANY)
            (S "No."
               (W 500)
               "Try again."))
  FAIL      (S "That's still wrong."
               "Try using the HELP system to figure out your problem.")
]
```

Except in sharply limited contexts such as mathematical courses where one asks for the solution to an equation, we discourage including genuine short-answer questions (such as, "What part of the brain governs involuntary motor functions?") in VOCAL lessons. The capability is there for writing a good short-answer question, using CAS to catch the exact desired answer and any common mispellings, WAS to cover any anticipated mistaken responses, and HINTL to provide a list of hints. But generally, the amount of care which would have to be taken to provide for all reasonable contingencies would just not be worth your while, and teaching assistants might find themselves swamped with students insisting that their answers should have been counted right by the computer. (Often a decision which would be accepted gracefully if coming from a human grader will be contested when it comes from a computer program -- perhaps justifiably, since the program cannot react to an unanticipated student response by looking at the question from a new angle.) These problems can be easily avoided with little loss of pedagogical value by rewriting the question in a multiple-choice format (for example, "Which of the following parts of the brain governs involuntary motor functions?").

4.4 Translation Questions

This type of question, which uses the TRANS opcode in answer analysis, is relevant only to courses which use the logic machinery. Usually, the question will be a request to translate an English sentence into one of the formal languages accepted by the proof checker. Some of these will use the Q opcode, and will either appear alone or be followed by a derive-class opcode using the symbolized sentence. Others will use the SYMB opcode (see [3], Section 2.14: Derive-class Opcodes).

The number of repetitions allowed should be between 3 and 5, depending on the complexity of the translation. Symbolic sentences are often rather long; there is a lot of room for making simple non-conceptual errors like typing a wrong letter or omitting some piece of the formula which one meant to include. The student should be given a decent chance to correct all these careless errors before failing the question. (The program helps out in this regard — when a mistype results in a string which is not a legal formula, the student is informed of this and the mistyped response is not counted as an attempt at the question.) But if a student cannot get the solution in 5 attempts, he probably has a conceptual problem, or is failing to take into account some basic feature of the sentence. Additional repetitions of the same question may only lead to frustration with the program, while revealing the correct solution, perhaps along with some comments on the important ideas in it, will enable the student to move quickly on to the next question with an increased understanding.

In translation questions, perhaps more than any other type of VOCAL exercise, care should be taken to inform the student <u>why</u> an incorrect response of his is not accepted. In particular, try to provide separate analyses for (1) formulas which are logically equivalent to the given English sentence but represent poor paraphrases, and (2) formulas which involve an error in logic. Translation of a student's incorrect response back into English, and presentation of a counterexample may both be helpful in this context. In addition, you may often want to use CAS to distinguish between the best translation, or the one which will be used in the rest of the exercise, and other acceptable responses. The following two examples illustrate these points.

```
[TEM "

Translate the sentence:  Some non-whites are disadvantaged.        %

Let  W(X) = X is a white

     D(X) = X is disadvantaged                                     %1


Symbolization:  (E X) ( NOT W(X) & D(X) )                          %2_
                FFFFFFFFFFFFFFFFFFFFFFFFF
     The sentence  NOT (A X) ( D(X) -> W(X) )  would be read as:   %3_
                   GGGGGGGGGGGGGGGGGGGGGGGGGG

'Not everyone who is disadvantaged is white', which is logically   %4_
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

equivalent to the original sentence, but is not a good paraphrase. %5
"
(Q
  INIT  ((S (T 1)
            "Now translate this sentence into predicate logic.")
         (T "
Type a formula of predicate logic."))
  R      3
  A      (TRANS " (E X) ( NOT W(X) & D(X) ) "
                " (E X) ( D(X) & NOT W(X) ) ")
  CA     (S ((T 2) (B F))
            "That's right.")
  WAS   ((TRANS " (E X) NOT ( D(X) -> W(X) ) "
                " (E X) NOT ( NOT D(X) OR W(X) ) ")
         (S "That's logically equivalent to the sentence"
            "you were supposed to translate,"
            "but it's not the answer we want."
            "Try expressing it in a simpler way.")
         (TRANS " NOT (A X) ( D(X) -> W(X) ) ")
         (S "Wrong."
            ((T 3) (B G))
            "Your response would be read as:"
            ((T 4) (B N) (T 5))
            "Not everyone who is disadvantaged is white."
            (W 500)
            "It's logically equivalent to the sentence"
            "you were supposed to translate,"
            "but it isn't a good paraphrase."
            "The two sentences would probably"
            "appear in very different arguments.")
         (W 2000) (E 3 4 5))
         (MATCHANY)
         (S "Wrong. Try again."))
   FAIL  (S ((T 2) (B F))
            "Nope, that's still not it.")
)]
[TEM "

Let     J      =  Joe          M     =  Mary
        B(X)   =  X is a boy   O(X)  =  X owns a car               %
                               O
        D(X,Y) =  X dates Y                                        %1


Translate:    The only boys whom Mary dates are those who own cars.%2

Translation:  (A X)( D(M,X) & B(X) -> O(X) )                       %3

"
(Q
   INIT  ((S (T 1)
            "Use these symbols to translate the sentence:"
            (T 2)
            "The only boys whom Mary dates are those who own cars."
            (W 500) (B O)
            "Be sure you type the letter $2 O"
            "and not the number zero!")
         (T "Answer by typing in a formula."))
   R     4
   CAS   ((TRANS "(A X)( D(M,X) & B(X) -> O(X) )")
          (S (T 3)
            "Very good.")
          (TRANS "(A X)( B(X) & D(M,X) -> O(X) )"
                 "(A X)( D(M,X) -> B(X) -> O(X) )"
                 "(A X)( B(X) -> D(M,X) -> O(X) )")
          (S "Yes, that's a correct translation."
            (T 3)
            "But we will be using a slightly different one in the
            derivation."))
   WAS   ((TRANS "(A X)( D(M,X) -> B(X) & O(X) )"
                 "(A X)( D(M,X) -> O(X) & B(X) )")
          (S "No, your symbolic sentence means:"
            "Mary only dates $2 boys who own cars,"
            "which is not quite the same thing."
            (W 500)
            "For example, the sentence on the screen"
            "leaves open the possibility that Mary dates $1 men"
```

```
                "regardless of whether or not they own cars.")
            (MATCHANY)
            (S "Sorry, that's not a correct translation."))
    FAIL    (S "Wrong."
                "Several translations are equally acceptable,"
                (B 3)
                "but this is the one we will use in the derivation.")
)]
```

4.5 Controlling Branching

The Q opcode can also be used to let the student directly control part of his path through the course, as in this VOCAL segment:

```
[Q
    INIT   ((S "The next exercise contains some additional material"
               "on non-commutative groups."
               "It's not used in the rest of the course,"
               "but you may find it interesting."
               (W 500)
               "Would you like to see this material?")
            (T "Type Y to see the extra material, or N to skip it."))
    HINT    (T "
The extra material consists primarily of common examples
of systems which behave as non-commutative groups.")
    CAS    [(AFFIRM)
            (S (COMMENT "fall through to next exercise") "Good."
               "It's nice to see a little intellectual curiousity.")
            (NEGATE)
           ((S (COMMENT "skip around next exercise")
               "Fine.  We'll go on to the next topic.")
               (GOTO 11 4))
            (MATCHANY)
           ((S "You must have accidently mistyped."
               "We're skipping the extra material,"
               "and going ahead to the next topic."
               "If you really meant to type a $2 Y,"
               "you can still hear more about commutative groups"
               "by looking at Exercise 3 of this lesson in Browse Mode.")
               (GOTO 11 4))]
]
```

If you are going to use questions in this way, it should be made clear to the students at the beginning of the course that typing ^H for a hint in this context will give a further description of the optional material. Notice that any reply is counted correct, so that the student's decision cannot adversely affect his cumulative correct-answer score, which is used to control non-voluntary branching. Notice also that the GOTO opcode occurs <u>outside</u> the accompanying Speak. Waits and compiler-specific opcodes like COMMENT are the only non-display-type opcodes which can occur within Speaks.

4.6 <u>Cascading Questions</u>

Although any type of question can form part of a CQ, there are some things which deserve to be said about effective use of this opcode. You may want to occasionally include a sequence of review exercises in your course; cascading questions are very useful in this context. The first 'tier' of a review cascade should be a question which, though not necessarily difficult, calls upon a broad spectrum of past material and contains enough choices to discourage random guessing. This will serve to weed out the students who need more review from those who can be allowed to move on to new material. The succeeding tiers should be simpler questions, written more to <u>remind</u> the student of material he may have forgotten than to test him on his retention. Two or three CQ's, each containing about 3 tiers, can thus form a review session which adjusts to the needs of different students.

Another good way to use CQ is in pairing a difficult exercise with an easier one, or an exercise which brings out an important new concept with a second one which uses the same concept in a similar way. This will (1) enable students who miss the first question to raise their morale by getting the next one right, and (2) promote retention of the relevant concept by allowing the student to use it immediately. Don't neglect this use of CQ! Often authors avoid using CQ very often because the increased number of questions makes a lesson seem too long, but actually very few of the 'back-up' questions will be seen by any one student, so the length of the course in student-hours does not increase very much when CQ's are used. Like detailed WAS analyses, any given segment of extra material will be seen by a small percentage of students, yet every student will see enough of them to improve his understanding of the subject. It's a case of quantity in the source file coming out as quality in the actual course.

Here is an example of CQ used in a review exercise:

```
[EXERCISE 10  "Review of Terms"
[AUDIO
(TEM2 "
Which of the following expressions are terms ?                        %1

  1)  Thomas Jefferson           7)  (A X) (X + Y = Z)                %_
  AAAAAAAAAAAAAAAAAAAA           GGGGGGGGGGGGGGGGGGGGG

  2)  is preposterous            8)  X + (Y + Z)                      %_
  BBBBBBBBBBBBBBBBBBB            HHHHHHHHHHHHHHH

  3)  6                          9)  every dog on the block           %_
  CCCCC                          IIIIIIIIIIIIIIIIIIIIIIIIII

  4)  X + 3 = 6                  10)  the current President of the USA%_
  DDDDDDDDDDDDDD                 JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ

  5)  P(Y)                       11)  X hates Y, and Y loves Z        %_
  EEEEEEEE                       KKKKKKKKKKKKKKKKKKKKKKKKKKKKKK

  6)  Y                          12)  (E X) ( X + 2 )                 %2_
  FFFFF                          LLLLLLLLLLLLLLLLLLLL

DEFINITION:  A term is an expression which denotes a specific         %
   individual, or does so when the variables in the expression
   are replaced by terms which do denote specific individuals.        %1
"
"
  Which of the following expressions are neither terms or formulas ?%11
"
(CQ
  [INIT  ((S (T 2)
            "In the last exercise,"
            "you identified the formulas on this list."
            (T 1)
            "Now find the $1 terms.")
         (T "
Type the numbers which precede the terms on the list."))
   HINT   (S (I 3)
             "Here is the definition of `term' again,"
             "in case you'd like to refer to it.")
   R       3
   A      (ALLANS "1" "3" "6" "8" "10")
   CA     (S "Good."
             (B A C F H J)
             "You found all five terms.")
   WAS   ((ALLANS "3" "6" "8" "10")
          (S "Almost, but not quite."
             (B C F H J)
             "You only listed four of the five terms here."
             (COMMENT "omitted:  George Washington")
             "Try to find the one you skipped, and remember that"
             "$2 constants are a type of term.")
          (ALLANS "1" "6" "8" "10")
          (S "Almost, but not quite."
             (B A F H J)
             "You only listed four of the five terms here."
             (COMMENT "omitted:  6")
             "Try to find the one you skipped, and remember that"
             "$2 constants are a type of term.")
          (ALLANS "1" "3" "8" "10")
          (S "Almost, but not quite."
             (B A C H J)
             "You only listed four of the five terms here."
             (COMMENT "omitted:  Y")
             "Try to find the one you skipped, and remember that"
             "$2 variables are a type of term.")
          (ALLANS "1" "3" "6" "10")
          (S "Almost, but not quite."
             (B A C F J)
             "You only listed four of the five terms here."
             (COMMENT "omitted:  X + (Y + Z)")
             "Try to find the one you skipped, and remember that"
             "terms can contain $2 operators.")
          (ALLANS "1" "3" "6" "8")
          (S "Almost, but not quite."
             (B A C F H)
             "You only listed four of the five terms here."
             (COMMENT "omitted:  the current President of the USA")
             "Try to find the one you skipped, and remember that"
             "anything which denotes a specific individual is a term.")
          (ANYANS "4" "5" "7" "11")
          (S "You listed at least one of the $1 formulas,"
             "which were identified in the last exercise."
             "An expression cannot be $2 both a term and a formula.")
          (ANYANS "9")
          (S "No, line 9 cannot be a term"
             "because it denotes a whole $1 group of individuals."
             "To symbolize this expression you would need"
             "a universal quantifier and a predicate,"
             "as well as a variable."
             (W 500)
```

```
                "Try to find the five terms.")
              (ANYANS "2")
              (S "No, line 2 is a $2 predicate, not a term."
                "Try to find the five terms.")
              (MATCHANY)
              (S "Wrong."
                "You must type the numbers ~~of~~ all five terms on the list."
                "Try again to find them."))
       FAIL   (S "No, you still haven't listed the five terms."
                (B A C F H J)
                "They are lines number: 1, 3, 6, 8, and 10.")
      ]
      [INIT  ((S ((OT 1 11) (U 2))
                "Finally, list the expressions"
                "which are neither terms nor formulas.")
              (T "
Type the numbers which indicate your answer."))
       R      2
       A      (ALLANS "2" "9" "12")
       CA     (S (B B I L)
                "Right!")
       WA     (S "Wrong."
                "There are three expressions on this list"
                "which are neither terms nor formulas,"
                "and if you were paying any attention to the last two
                 questions,"
                "it shouldn't be hard to find them."
                "Try again.")
       FAIL   (S "Wrong again."
                (B B I L)
                "The answer is lines 2, 9, and 12.")
      ]
)  (COMMENT "end of cascading question")
)
]]
```

References

1. Theodor H. Nelson, "No More Teachers' Dirty Looks", Computer Decisions, September 1970.

2. Suppes, Patrick, and Morningstar, M. "Computer-Assisted Instruction". Science, Oct. 1969, v. 166, pp. 343-350.

3. Hinckley, Micheal, et. al. VOCAL: Voice Oriented Curriculum Author Language. Stanford, CA: Stanford University, Institute for Mathematical Studies in the Social Sciences. Technical Report No. 291.

4. Beard, Marian H., et. al. Comparison of Student Performance and Attitude Under Three Lesson Selection Strategies in Computer-Assisted Instruction. Stanford, CA: Stanford University, Institute for Mathematical Studies in the Social Sciences, Technical Report No. 222.