DOCUMENT RESUME

ED 122 753 IR 003 361

AUTHOR Laymon, Ronald
TITLE ENIGMA, CAI-CMI for Introductory Logic: Some of Its Abilities with English Sentences.
PUB DATE Jan 76
NOTE 7p.; Paper presented at the Association for the Development of Computer-Based Instructional Systems Winter Conference (Santa Barbara, California, January 26-29, 1976)

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage
DESCRIPTORS *Computer Assisted Instruction; Computer Graphics; *Course Descriptions; Higher Education; *Logic; Philosophy; Program Descriptions; Testing; Tutorial Programs
IDENTIFIERS Computer Managed Instruction; Drill and Practice Programs; *ENIGMA; Natural Language Programs; Ohio State University

ABSTRACT

The philosophy department of the Ohio State University began development of a computer-tutorial program, called ENIGMA, in 1972. The aim of the course was to help students to use various logical tools in the analysis of everyday arguments by giving drill-and-practice sessions, testing, and grading examinations. Part of ENIGMA is the propositional argument program which is able to abstract the relevant syntactical form from the student's natural language response and check the response for logical mistakes, gross spelling errors, and violations of propositional integrity. This program can be used in a test as well as a tutorial format. A program for categorical syllogisms has a feature offering the capability of using Venn diagrams to determine validity. The programing language for ENIGMA is Coursewriter II; it is timeshared on an IBM 370/158. It is expected that by spring of 1976 approximately 1,000 students per quarter will use the program. (JY)

Ronald Laymon
Assistant Professor
Department of Philosophy
The Ohio State University
Columbus, Ohio 43221
Phone: 614-422-5094

Title: ENIGMA, CAI-CMI for introductory logic: some of its abilities with English sentences

Introduction

In an effort to improve and personalize the instruction of introductory logic courses, the Philosophy Department of The Ohio State University commenced the development of a computer-tutorial program, ENIGMA, in the spring of 1972. The programming software is nearly completed, and full implementation, for approximately one thousand students per quarter, is scheduled for spring 1976. The programming language is IBM's Coursewriter III, version 3 as augmented by the Ohio State set of functions; the system timeshares on an IBM 370/158. When in full operation the program will utilize thirty-one Hazeltine 2000 terminals. Average student time on the terminals will be one and a half hours per week.

The general aim of the introductory logic course is to impart to the student the ability to use various logical tools in the analysis of everyday arguments. The orientation of the course is not toward the type of symbol manipulation appropriate to mathematical logic. The computer will be used to provide tutorial, i.e., supervised drill and practice, for students who will have already been exposed, via lectures and reading, to the course material. In addition to supervising students' drill and practice, the computer will refer students with persistent difficulties to live teaching staff. Finally, the computer will administer and grade the course examinations.

The content of the course includes both deductive and inductive logic. Some of its distinguishing features are: its ability, in certain lessons, to accept sets of English sentences as the appropriate student response; its ability to randomly generate and solve problems; its use of computer graphics, and finally, its extended managerial capabilities.

A basic goal of the course is to enable students to:
(1) rewrite ordinary English arguments, or purported arguments into the appropriate logically standard English form, where this standard form will reflect the relevant logical properties of the given argument,
(2) abstract from this standard English form the essential symbolic or syntactic form of the argument
(3) evaluate the abstracted form for validity (or in the case of inductive arguments for strength of inductive support)

Given this basic goal, it was determined that at least some of the computer tutorials should be capable of accepting and adequately analyzing sets of English sentences as the normal student response.

Propositional Argument Program

I will now describe our propositional argument program. Similar programs exist or are in the developmental stage for categorical arguments of syllogistic form as well as for more general but simple arguments of quantificational form. From the point of view of ordinary predicate calculus (the normally used canonical form) the most basic argument forms are propositional; i.e., the logical properties depend on the ways the component sentences are connected and not on their internal structure. These argument forms also provide a natural starting place for the educational process. (There are, of course, several important qualifications to be made here, but this is not the appropriate place.)

The propositional argument program begins, for the student, by presenting (in ordinary English) a text of a propositional argument with one conditional and one non-conditional premise. For example,

> Socrates didn't accept the benefits of the state, thus he shouldn't obey the law, because he should obey the law only if he had accepted those benefits.

The immediate student task is to:
(1) distinguish premises from conclusion
(2) translate all conditionals into a standard if-then form
(3) eliminate premise and conclusion markers such as 'because,' 'since,' 'hence,' and 'therefore'
(4) eliminate logically irrelevant expressions such as those of emphasis

The student distinguishes the premises from the conclusion by entering a three line response, with the premises as the first two lines, and the conclusion as the last line. (It is not necessary that the student put the conditional premise first.)

After the student completes his three line response, the program begins its analysis by checking for some elementary and easily spotted mistakes. These fall into three categories:
(1) Minor errors such as the omission of 'then' from 'if-then' statements; the failure to eliminate conclusion and premise markers.
(2) Non-translation of the conditional premise into an 'if-then' form
(3) The use of double or iterated negation

Mistakes of the type (1) are reported by ENIGMA to the student, but the student is not forced to rewrite his answer since he may still have the logic of his answer essentially correct. Students, though, must rewrite mistakes of the type (2), since these mistakes signify that conditional statements have not been properly translated into standard if-then form. Students are also asked to eliminate type (3) mistakes, iterated negations.

Originally it was also required that students replace all pronouns with their antecedents; later this requirement was softened so that ENIGMA only mentioned such lapses. Our experience with the program has led to the more radical modification that we now encourage students to use pronouns. This move is natural and saves typing time. ENIGMA will require the substitution of antecedents only in those cases where ambiguity results or where an improper pronoun has been used.

ENIGMA will next abstract the relevant syntactical form from the student response. If this abstraction were not made, all the possible errors and correct answers for each text would have to be separately generated and stored. Since, however, we need only check for syntactic or logical features, it does not matter if the semantic content specific to each individual text is lost via the process of abstraction.

The abstraction program identifies the subjects, verbs, and objects of the propositional components of the student's answer. The program will tolerate examples that have propositional components with the same or different verbs, the same or different objects, and with the same, different, or no objects. In addition, account has been taken of the relevant syntactical differences between sentences with transitive versus copulative verbs. Finally, variations of English tense structure are handled by scanning only for verb roots (unless logically important modalities are expressed with helping verbs).

The programming necessary for the text specific part of the abstraction can be entered by means of one of ENIGMA's macros. All that needs to be entered is an ordered list of subjects, verbs and objects with the usual Coursewriter '&' and '*' symbols to indicate acceptable misspellings. We have at Ohio State developed a modified function edit routine that will accept these symbols.

Once the student's response is symbolized, two sorts of mistakes are checked for: a) "Gross" spelling mistakes; b) Violations of propositional integrity.

a) The symbolizing part of the program is designed to recognize typical misspellings. For example, "Socratis", "Socrate", "Socrats" all count as "Socrates". Should the student, however, make a spelling error not ordinarily picked up in our program, this "gross" misspelling is located by the programming immediately following the abstraction section. The student is informed of the location of his spelling error and is asked to correct it. (With the cathode ray tube terminal the student need not retype his entire answer but can selectively correct only that part which has been misspelled.) This part of the program will also pick out and locate radical substitutions for correct answers. For example, substituting "Plato" for "Socrates".

b) "Violations of propositional integrity" refers to the mixing of the parts of the propositional components of the argument so that those component propositions lose their identity. An example of such a "violation" would be writing:

If John quit Martha, then John loves politics,

instead of

If John loves Martha, then John quit politics.

Mistakes of both types (a) and (b) will usually be either mere slips on the student's part of deliberate attempts to "beat" the computer program. In either case the student is given a computer response specific to the error (along with, in some bases, a note on the difficulty of determining student motives).

Finally, the computer checks the student's answer for any logical mistakes. (This forms the core of the propositional program.). Logical mistakes, for the purposes of this tutorial, are (1) confusing premises and conclusion; and (2)

confusing negations or converses for correct answers. In addition, the proper ordering and number of negations is is checked for in the conditional premises. (The program, incidentally is not confused by the substitution of contrapositives, i.e., writing a statement of the form 'if not-q, then not-p', instead of its logical equivalent, 'if p, then q'.) Comments appropriate to each of these mistakes are given to the students. Persistent confusion on the student's part branches him to remedial section or off the terminal to "live" instructional staff.

The ENIGMA propositional argument program has some additional features.
(1) If the student makes a serious logical mistake he is, after he has gotten it correct or had it corrected, given the option of rewriting his answer for succeeding lines.
(2) If the student cannot provide the correct answer, he may after a minimum number of tries opt for ENIGMA to supply the correct answer. ENIGMA need not be told the correct answers by instructors. All that needs to be entered via the appropriate macro is an ordered set of subjects, verbs and objects. ENIGMA will always supply the student with the correct answer for any of the possible 32 propositional argument forms instantiated by the given text. (More on the process of text and problem entry is given below.)

After the student successfully completes his rewrite of the given text into a logically standard English form he is asked to symbolize answer as a propositional argument. For example:

If not p then q
q
not p

(Note: This symbolization differs from that used by ENIGMA to recognize student response. ENIGMA's abstraction program breaks student responses down to the parts, i.e., subjects, verbs, objects, negations, of the component propositions.)

Should the student make a mistake here, ENIGMA will supply the correct answer along with the proper substitution for the variable letters used.

The anticipated correct symbolic form is automatically adjusted by ENIGMA to be a function of the student's standard English response. Thus if the contrapositive, for example, is entered, the correct symbolic form is adjusted accordingly. ENIGMA will also recognize eliminable occurrences of 'not.' ENIGMA will accept their explicit mention by students but will remind students of the existence of either simpler or more familiar forms.

Finally, the student is asked to determine the validity of the argument form. With respect to these simple propositional forms this process is primarily one of recognition.

Entering a Problem Text

Entry by an ENIGMA user of a new problem text occurs in essentially two steps. First, one must enter in an ENIGMA macro the subjects, verbs and objects (if they exist) of the component propositions. The misspelling code is also to be entered here. Second, one can now enter up to 32 variations of propositional

arguments that are constructible from the given ordered components. All that needs to be done is to enter the ordinary English text along with a six digit code.

(For any two different statements p and q, there exist 128 different propositional arguments with one conditional premise, one non-conditional premise, and a non-conditional conclusion. (There are eight possibilities for the conditional premise, if p then q, if not p then not q, if p then not q, if not p then q, plus their converses; four possibilities for the non-conditional premise, p, not q, q, not p, and those same four possibilities for the conclusion.) If the argument forms are restricted so as not to include propositional functions (i.e., negation and not negation) of the same proposition as both non-conditional premise and conclusion, the number of variations reduces to 64. Further restriction of these forms to the classic modus monens, modus tollens, and the fallacies of affirming the consequent and denying the antecedent, reduces this number to 32. Finally, if the conditional premise indicates, for example, a causal connection, the number is reduced to around 16. Now while only these last two sets of 32 and 16 are used in the tutorial, the program as it now stands can be used for all 128 variations. This enormous extra flexibility means that the program can be expanded in rather obvious ways so as to incorporate more complex forms of propositional arguments.)

I have in the developmental stage a program that will directly generate ordinary English text directly from the list of ordered propositional components. All told, approximately 30,000 variations are possible. There are first of all the 32 basic argument forms; next there are 6 different orderings of premises and conclusion; then there are 8 different forms for the English conditional. This gives 2048 possible forms. Finally, superimposed on these forms are between 8 to 12 variations of English premise and conclusion markers (along with expressions of emphasis); the result is about 30,000 variations of English text.

Test Format

The propositional argument program can also be used in a test as well as a tutorial format. The test format limits students to single attempts (though the program will allow for the correction of "gross" spelling mistakes before grading begins.). A simple correct/incorrect response is given. In the case of an incorrect response, the correct answer is supplied.

Graphics; Venn Diagrams

The program for categorical syllogisms is very similar to that used for propositional arguments. It has, though, the additional feature of offering students the capability of using Venn diagrams to determine validity. Quasi-graphics are possible using the normal character set and the Ohio State function CRT routines.

Tutorial-Test Combinations

While, as I have already noted, ENIGMA is now used in a combination tutorial-test mode, we are generalizing our managerial-operations system to allow for three general formats: (1) combination tutorial-test; (2) exclusively tutorial;

- (3) exclusively test, with an option for a pre- and post-test mode.

Data Stored

Because ENIGMA will be used at Ohio State with approximately 1000 students per quarter, we have opted for an immediate on-line data retrieval system. This system avoids the usual delays associated with off-line tape systems. Very detailed and extenive data can, of course, also be stored and sorted off-line.

The data stored on-line for the propositional arguments program is: (1) the number of problems completed; (2) the overall numeric grade; (3) the corresponding mastery level (these levels are under the control of course instructors); (4) success scores on the rewrite into logically standard English; (5) raw data on logical errors separated into categories of premise/conclusion confusion and mistranslation of conditionals; (6) finally, a measure is kept of student entry difficulties, e.g., gross spelling errors.