COSC 460

Research Project

Department of Computer Science

University of Canterbury

NEW ZEALAND.

A PREPROCESSOR TO THE STAF LANGUAGE

LEE HOCK HIN

1980

TABLE OF CONTENTS

APPENDICES

PAGE

ACKNOWLEDGEMENT

I would like to thank Dr R. Maclagan of the Chemistry Department for his help and supervision in my project. His interest and his knowledge of the STAF system has been invaluable to the progress made.

I would also like to thank the STAF users for their helpful comments and suggestions about the preprocessor.

1 INTRODUCTION

1.1 Brief Background

The term Computer Assisted Learning (CAL) covers the area of using the computer to assist in the learning situation. CAL is known as CAI (Computer Assisted or Aided Instruction) in the United States.

There is a variety of methods by which a teacher can use to produce computer assisted learning programs. The Science Teachers' Authoring Facility (STAF) is available at the University of Canterbury for the authoring, or writing, of CAL programs. It has a specialised high level authoring language, the STAF language [1].

1.2 The Aim of the Project

The aim of this project is to design and implement an interactive Preprocessor to the STAF language. The Preprocessor will give the teacher an easier alternative to produce STAF CAL programs as the STAF language is not suitable for non-programmers.

1.3 Resources Available

This is the first project of its kind at Canterbury, with no previous research work to follow up. As I had no background on computer assisted learning, an initial objective was to obtain the necessary background. The next section gives the findings of my search through the literature.

2 BACKGROUND ON CAL

2.1 General Information about CAL

The distinctive characteristic of computer assisted learning is its responsiveness. The student maintains a conversation with the computer. The computer replies appropriately after analysing the student's response. The CAL program gives the student extra help where needed.

CAL is used in several modes of instruction. These include Simulation, Games, Drill & Practice, Quizes and Tests. True simulation and games are more sophisticated and are different from the CAL programs mentioned in this project.

CAL is also classified by its degree of use. CAL lessons of half an hour to one hour duration are used to supplement work done in the regular classrooms. On the other hand, there are courses which rely entirely on CAL. Distance learning, similar to correspondence courses, can rely heavily on CAL. The CAL programs which the preprocessor will produce is more of the first type.

The student communicates with the computer through a workstation. Workstations can vary from simple printer terminals or CRT terminals as available in computer installations to specialised and sophisticated devices. One such sophisticated device is shown in Figure 1. It is quite similar to the workstations used in the PLATO system [2]. But the heart of every CAL system is the program which makes up the lesson. This program is generally known as COURSEWARE.

DESIRABLE FEATURES FOR AN INSTRUCTIONAL TERMINAL

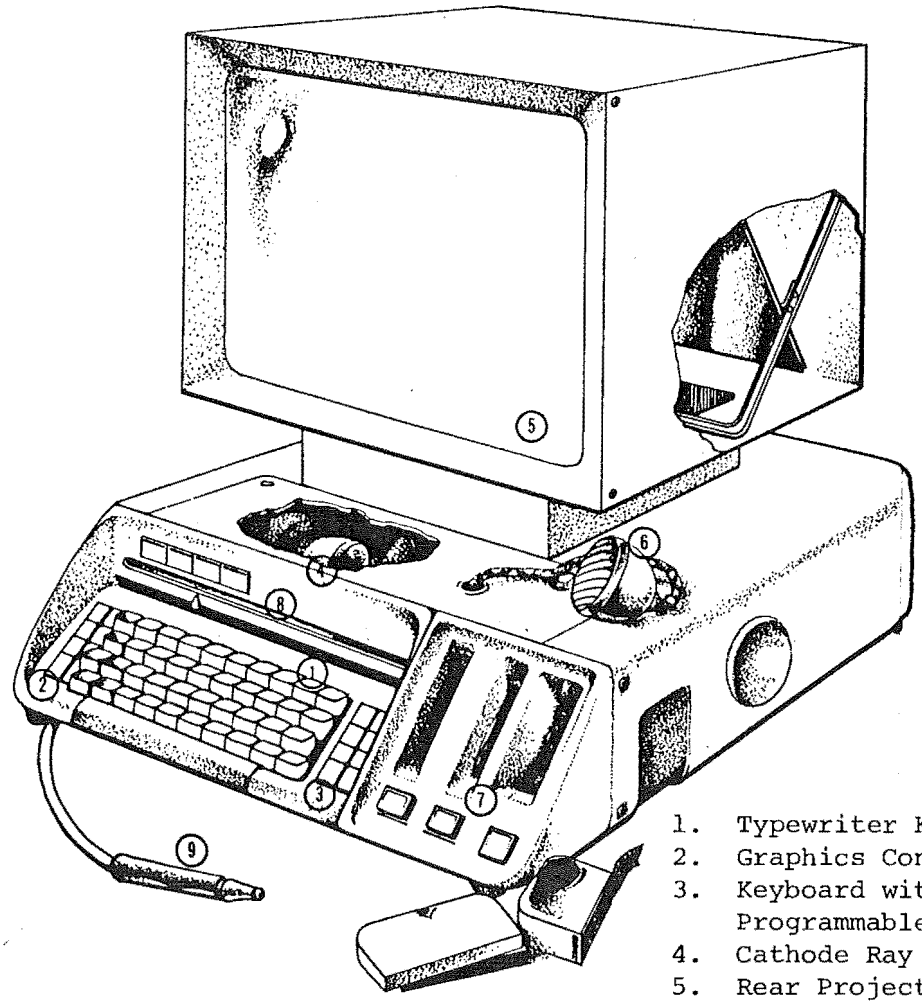

Figure 1

2.2 Courseware

The development of courseware is of critical importance as the effectiveness of CAL depends directly on it.

CAL programs have a structure which is frame (several examples of frames are given in section 2.4.3 - The STAF language) or node oriented. Each frame has two main components, the text displayed to the student and the answer-matching scheme. The answer-matching scheme will route the student to the next appropriate frame after analysing the student's answer.

Dean [3] says that every CAL program consist of two distinctive parts - The Instructional Design and the Course Content. The Instructional Design or Teaching Strategy involves the method of presenting a topic and the decision-making at each frame. This is best done by a pedagogical expert. The Course Content has to do with the subject matter the teacher wants to present. It is worthy to have these two parts separated and done by the appropriate person, but this does not appeal to the individual teacher.

2.3 CAL Authoring Systems

Writing a CAL program is a lengthy and non-trivial job. The production of quality courseware involves a cycle where the CAL program is improved using student feedback. To assist the CAL author, many authoring systems have been developed. Five major methods are discussed.

1) Teams of Teacher, Pedagogical Expert and Programmer

This is generally agreed as the best method to produce quality courseware. Figure 2 depicts such a system. It requires several people and more of their time as they need to interact among themselves. Departments with modest amounts of resources cannot support such teams.

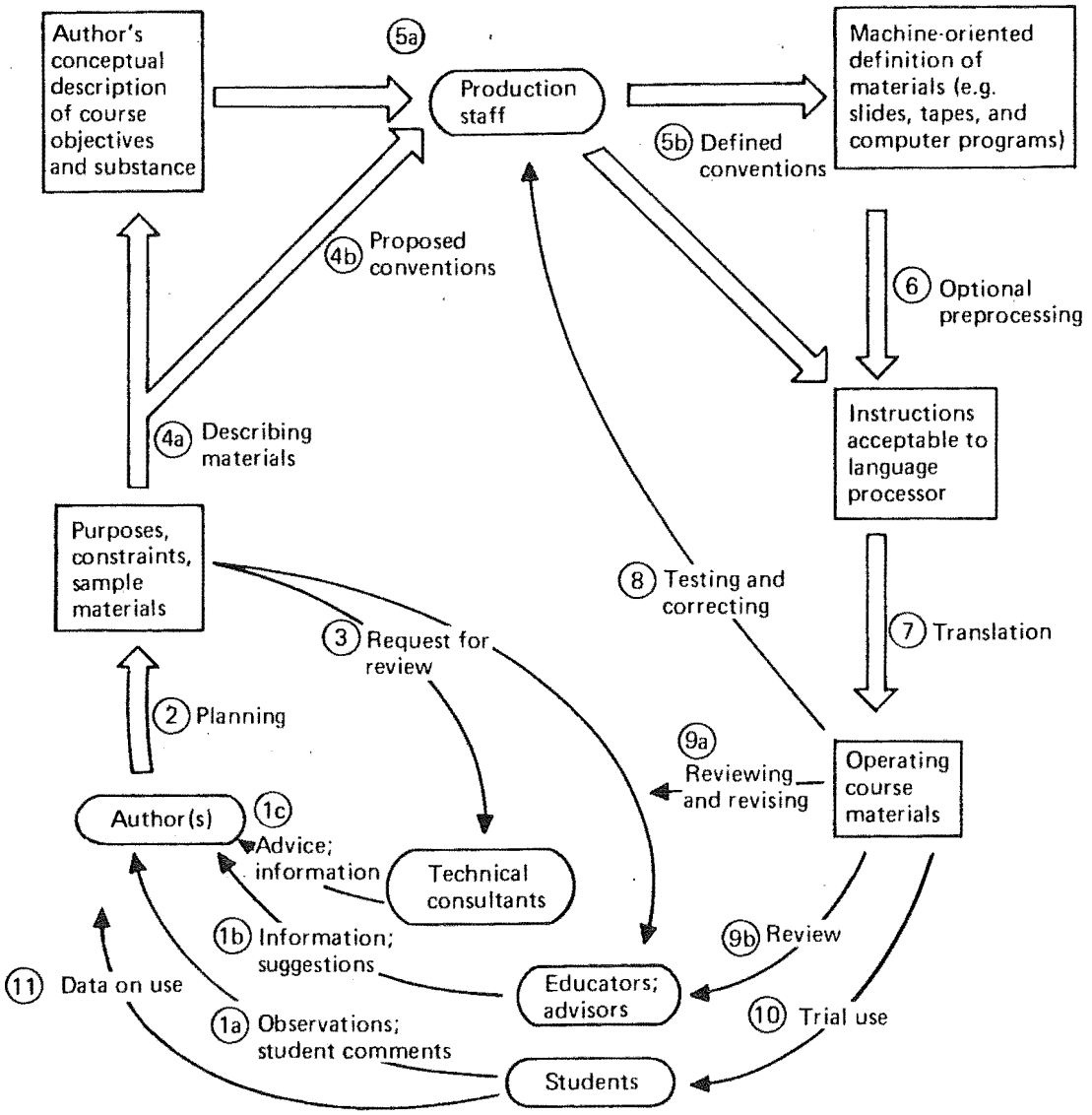

Fig.2. One representation of authoring activity.

2) Special Languages for CAL authoring

Special languages were developed specifically for CAL authoring. They make CAL authoring less tedious but are usually too cryptic for a non-programmer. They also have the disadvantage of being machine dependent. Examples of these languages are COURSEWRITER by IBM, TUTOR which is used in the PLATO system and STAF.

3) General Purpose Languges

Programming languages like BASIC, APL, FORTRAN and PASCAL are also used for CAL authoring. They are used either because the

teacher knows the programming language or because the language is available. General Purpose Languages are not suitable as they make CAL authoring too tedious.

4) Interactive Prompters

Prompters elicit the necessary information from the teacher through prompts and questions. They are easy to use but have a tendency to produce CAL programs which are lacking in instructional design. The preprocessor of this project comes under this method. An example of prompters is COURSEMAKER III.

5) Editor-Authoring Systems

These systems utilise the power of text-editors to produce CAL programs. They are not flexible and are of limited use. Test-Gen, an Editor-Authoring System, specialises in generating tests.

In writing courseware, the author should be wary not to display too much text. Together with a lack of instructional design, it turns a CAL program into a "programmed textbook". Such programs are characterised by the lack of responsiveness.

2.4 STAF Background

2.4.1 General Background

The STAF system was implemented as part of CALCHEM (Computer Assisted Learning in Chemistry) which is a project of the U.K. National Development Programme in Computer Assisted Learning. STAF itself is a version of the Leeds Author Language developed at the University of Leeds.

Knowledge and appreciation of STAF is necessary for the design of the preprocessor. The STAF CAL program is interpreted. The student interacts through a CANDE terminal linked to the B6700. Only text and numeric values are exchanged between the STAF program and the student. True graphics is not supported on either the B6700 system or the STAF facility. The student should also obtain a copy of the dialogue on a printer for future reference. Figure 3 gives the graphic representation of the system. A statistics file of the lesson is also produced for the teacher.

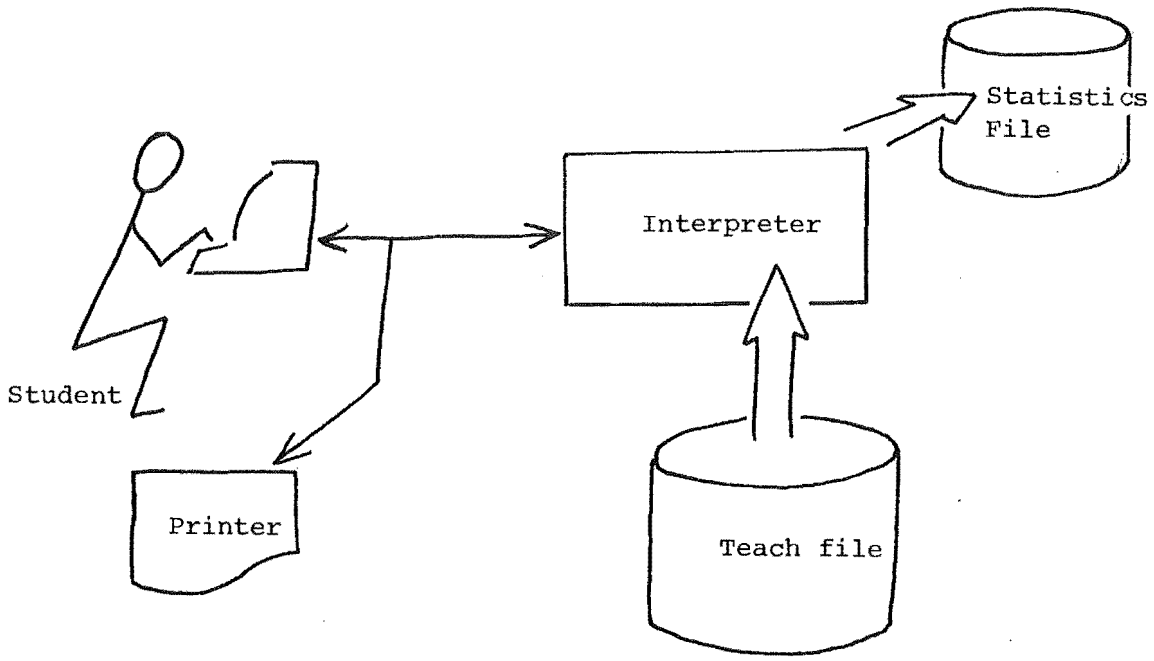

Figure 3 An Example of a CAL System: The STAF System

2.4.2 The STAF authoring system

The STAF authoring system is shown in Figure 4. The initial draft of the CAL programme is usually done on punched cards by the data preparation staff at the Computer Centre. The source program is fed into the machine and stored on disk. The source program is checked for syntactic errors by the Analyser. An object file, the teachfile, is created if there is no syntax errors. This teachfile is run on the Interpreter to test the logic of the CAL program. The teacher is expected to use the CANDE editing facility to correct syntactic or logical errors found in the source program. Both the STAF language and the CANDE commands are not easily learnt.

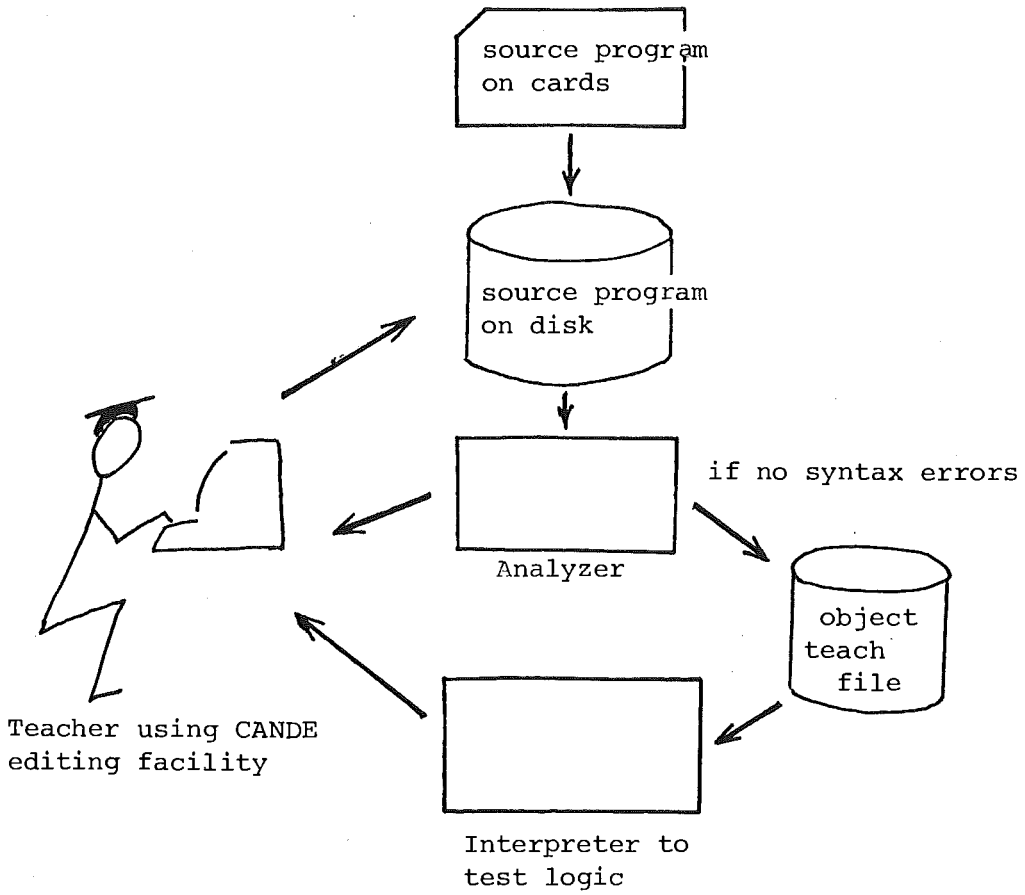

Figure 4 THE STAF AUTHORING SYSTEM

2.4.3 The STAF language

A complete STAF program can be expressed as a directed graph* with frames linked by routings. Each frame can have text to be displayed and answer-matching schemes with their consequent routings.

Every frame in a STAF program has at least two parts, the frame header and the final routing. The final routing is taken when the student's answer matches none of the anticipated answer-matching schemes.

Figure 5 gives three different examples of frames. The first frame is made up of the frame header (Line 1100), the text to be displayed (Line 1200) and the final routing (Line 1300). Frame headers begin with a "#" whilst routings being with a "!".

```
FRAME 1   1100 #M01;0*
          1200 YOUR CAR IS SKIDDING BECAUSE OF THE ICE ON THE ROAD.
          1300 !Q02;
          1400 #A01;0*
          1500 @0 SKID$SKD !B02;
FRAME 2   1600 %0 PUNCTR !B03;
          1700 %0 STEER WHEEL BROKE$BRK !B04;
          1800 !$LTCX01;N01$NR;M01$SR;Z01A01*
FRAME 3   1900 #N01;0* !$SR;Z03M01*
```

Figure 5 EXAMPLES OF STAF FRAMES

The second frame (Lines 1400 to 1800) has three answer-matching schemes. The matching is performed on a word basis. The routing at the end of an answer-matching scheme is taken if the student's response agrees. For more details please refer to the STAF manual [1].

Subroutines are prefixed with a "$". A subroutine or a string of subroutines can be used in a routing. The final routing of the second frame (Line 1800) is a string of 3 subroutines. Subroutines can manipulate and test counters (numeric variables) or alter the flow of control in the STAF program.

* For readers without any knowledge on directed graph, reference [5] gives the basic definition.

The third frame shows that it is not essential to have text or answer-matching. Figure 5 is a portion of the CAL program in appendix B which was generated by the preprocessor from the sample run in Appendix A. An idea of what Figure 5 represent can be obtained from Appendix A.

As can be seen, the STAF language is cryptic and not suitable for the non-programmer. I too had difficulties as there was no standard structure or construction to follow. This lack of structure gives the teacher unlimited flexibility. The features of STAF are powerful and suited for CAL programming. The answer-matching, if carefully used, can be very powerful.

3 DETAILED AIMS OF THE PROJECT

After the background research, I decided on the following aims for the project. These aims are given in the order of decreasing importance.

3.1 Construction of a Working Interactive Preprocessor to the STAF language

The preprocessor, by its very nature, will have certain desired attributes. It will remove the teacher from the cryptic details of STAF as the interaction is in simple English. It will decrease the time and effort needed for a teacher to transform his ideas into a working CAL program. This is because the preprocessor is always available and the CAL program produced will be free from syntactic errors. It also frees the teacher from the inherent house-keeping details involved with STAF programs, especially the labelling of frames.

3.2 Easy-to-Use and Easy-to-Learn

Since the preprocessor prompts a user along, it seems desirable to have the preprocessor able to prompt a teacher with no CAL experience along. Though CAL programs produced by the inexperienced teacher could be lacking in instructional design, the teacher could gain valuable experience from it. This would encourage teachers to produce CAL programs.

3.3 Unrestricted Potential of the STAF Language

The full potential of the STAF language must be available to the teacher who has CAL authoring experience. Many prompters have been criticised for their lack of sophistication and simple-mindedness because it hinders the production of quality courseware. Sophistication seems to be in direct conflict with the aim of making the preprocessor easy-to-learn and use.

3.4 Editing and Correcting Facility

CAL authoring is an iterative process in which a CAL program is improved using feedback from students. It is therefore necessary that STAF CAL programs produced by the preprocessor can be edited in a similar manner to which they were generated.

4 RESEARCH EFFORTS AND DEVELOPMENTS

There are several aspects to this project. The first was to obtain sufficient background. Other major aspects are now described, in the order which I confronted them.

4.1 Defining the Structure of a STAF program

The STAF language, by itself, has little structure but the preprocessor needs a definite CAL structure to build itself from. This structure is closely related to the instructional design and must be general so as not to restrict the flexibility and power of STAF.

4.1.1 Options and Alternatives Possible

Most CAL programs can be broken down into a series of modules, each with one entry point and one exit point. The instructional design, and therefore the structure, within each module is independent of other modules. The structural definition can therefore be narrowed down to within a module.

By doing so, I have restricted the authoring process to be breadth first. Breadth first authoring produces the program on a level by level basis, completing all the details at each presentation before going to the next. The other approach, depth first authoring, completes the direct path from the start of a CAL program to its end before looking at the side issues.

A module would present a question or problem to the student and have all the frames that are related to this problem. Some student answers will receive simple replies like 'CORRECT'. Students giving incomplete answers should be helped to arrive at the complete answer. This involves a teaching strategy.

A set of predefined teaching strategies could be supplied but is not ideal as this method is not flexible. Another option is to use sub-modules, that is a module within another module. This produced an implementation problem as frame labelling is limited in STAF. Also, there is no fixed structure and a teacher using several levels of sub-modules is likely to get lost. Defining the teaching strategy recursively as needed for nested sub-modules can be difficult.

Another related problem is unanticipated responses. The student could be asked to try again or directed to a teaching strategy which gives assistance.

Both these problems were left aside as it involved teaching strategies and I had insufficient knowledge on them. Finally, after much deliberation and in close consultancy with the STAF users, a basic structure was defined. This structure is given in Figure 6 and is described in the next section.

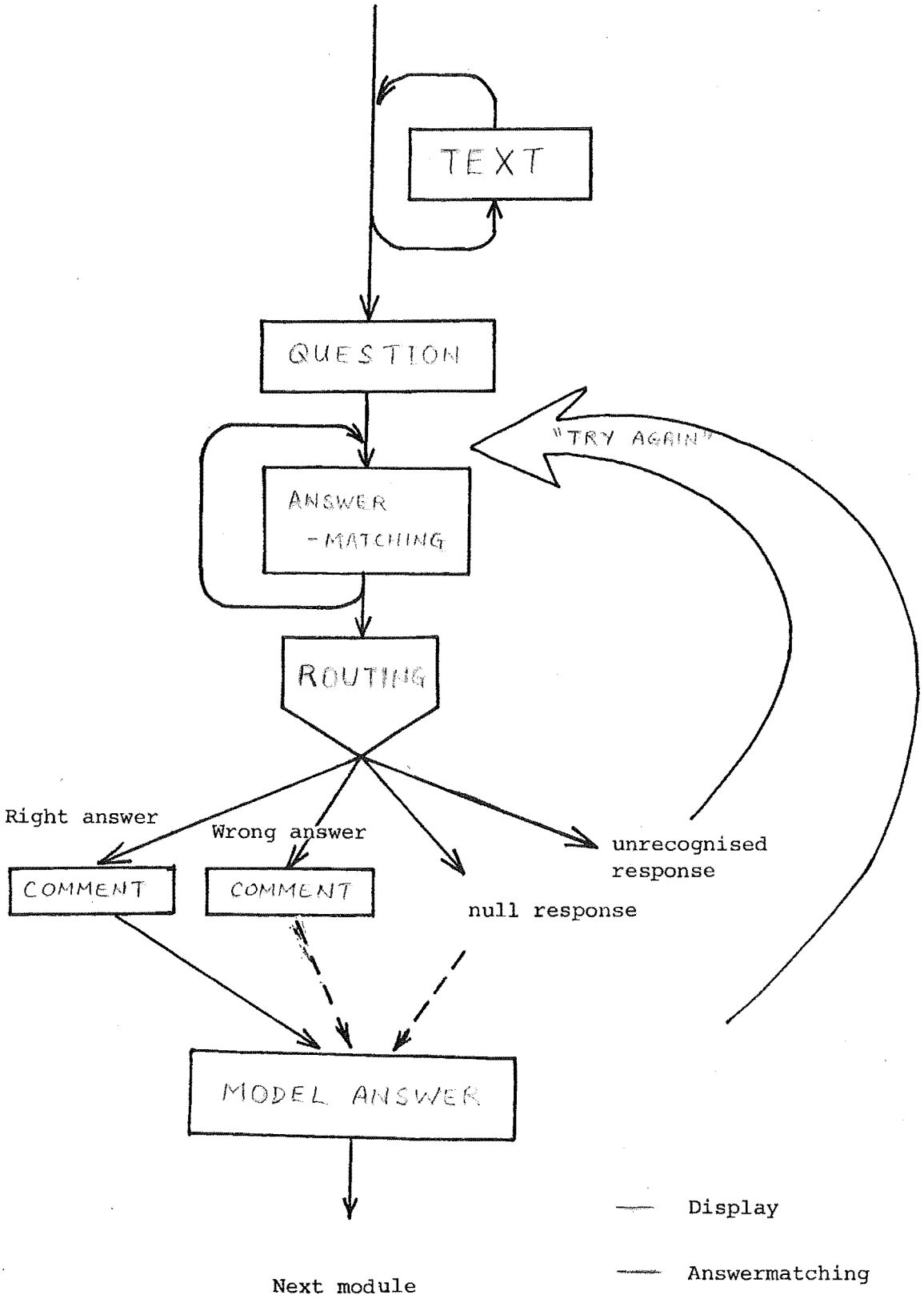

Figure 6 The Basic Module

4.1.2 Description of the Structure Chosen for a Module

The Basic Module is made up of two components: text displayed to the student (in blue) and answer-matching schemes with their routings (in red).

Before the QUESTION is presented, there can be a few pages of explanatory or descriptive TEXT. One segment or page of text is shown to the student at a time. The student types in a Linefeed when he is ready for the next page.

After the QUESTION is displayed, the CAL program waits for the student's response. ANSWER-MATCHING schemes are described to analyse the student's response. Student's responses are grouped as Right Answers, Wrong Answers, Null Response or Unrecognised Responses. All have simple replies and there is no provision for implementing teaching strategies as I was looking for the solution to this problem. Because the answer-matching is on a word basis, the complete MODEL ANSWER is always given to confirm the student's answer.

Right Answers are those which the teacher accepts as correct. There can be a short comment like 'CORRECT' for a right answer before displaying the model answer.

Wrong Answers can have a comment, after which the student is either given another attempt at the problem or routed to the model answer. If the student is given a second attempt the comment could contain a hint.

A Null Response is when the student enters a blank line. The teacher has a choice of either directing such a student to the model answer or giving the student one last attempt.

An Unrecognised Response is one which does not match any of the anticipated answers. The student is asked to try again. A limit of

four attempts is imposed as too many unsuccessful attempts can make the student impatient.

4.2 The Interactive Aspect of the Preprocessor

The next important design is the interactive aspect of the preprocessor. The dialogue with the teacher must be clear and coherent to the extent that a teacher with no CAL programming experience can use the preprocessor.

Much of the interactive design was developed in line with other interactive programs.

Figure 7 shows a portion of the dialogue between the teacher and the preprocessor. All messages from the preprocessor have the double arrowheads symbol ">>" at the start of the line (lines 2 to 5) to differentiate them from the messages originating from CANDE.

The teacher is prompted for his response by two means. When a "<?>" is printed at the start of a line by the preprocessor, it is ready to accept multiple lined responses from the teacher. A "#" immediately after the "<?>" terminates the input. This is shown in lines 6 and 7 of Figure 7.

The other mode of input occurs when the teacher is asked a question by the preprocessor. This is indicated by the cursor remaining at the end of the query. A YES/NO response is expected. The abbreviations Y and N can be used. This type of prompt is shown in line 3 of Figure 7.

The teacher can easily tell by either means whenever his response is requested. In Appendix A, other aspects of the preprocessor's interaction are described with reference to a sample preprocessor run.

```
1
2     >> LIGHT
3     >> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? Y
4     >> GIVE ALL ALTERNATIVES, ONE TO A LINE.
5     >> MAXIMUM OF 5 LINES.
6     <?>GEAR
7     <?>#
8
9     >> ANY MORE TOTALLY WRONG ANSWERS? N
10
11    >> CHOOSE ACTION TO TAKE WHEN STUDENT ENTERS A BLANK LINE
12                        0. GIVE MODEL ANSWER AND GO TO NEXT QUESTION
13                        1. ASK STUDENT TO TRY AGAIN
14    <?>1
15
```

Figure 7 AN EXAMPLE OF THE DIALOGUE WITH THE PREPROCESSOR

4.3 Potential of STAF is not Restricted

Another aspect of the project is that the preprocessor should not restrict the potential of STAF. This is an aim of the project as described in section 3.3. The preprocessor could limit the power of STAF in two ways, by restricting the structural flexibility of STAF or by limiting the features of STAF.

The structural flexibility involves the ability to implement various instructional strategies. This depends totally on the structure of the module used by the preprocessor. This has been discussed in section 4.1.

The features of STAF can be grouped into three types. They are:

i) the Display of Text,

ii) the Answer-Matching and Routing schemes

and iii) the use of Subroutines.

When displaying text, reserved characters must be prefixed with a double quote. This is automatically performed by the preprocessor.

Blank lines and paged output are possible. The only restriction is that a "#" cannot be the first character of a line. (A "." could be used instead, as it is unlikely to occur at the start of a line, but is less obvious to the teacher).

Answer-matching is performed on a word basis. There are three parts to each answer-match; the strictness of match, the words to be matched and the consequent routing. Because the syntax of the answer-matching schemes is well defined, it is implemented without any restriction.

The last feature, the use of subroutines has not been implemented.

5 RESULTS AND SUMMARY

The tangible result of this project is the working preprocessor. It is written in Burroughs Extended ALGOL. A reasonable amount of effort and work has been expanded in the design and implementation of the preprocessor. As it is an interactive program, the features of the preprocessor are best illustrated by a run. An example of a run together with comments is given in Appendix A. Appendix B gives the corresponding STAF program generated.

In summary the preprocessor is easy-to-use, requiring no experience. Terms used by the preprocessor must necessarily be understood and are best conveyed through a sample preprocessor run.

The full capabilities of STAF's text display and answer-matching schemes are available to the preprocessor user. The STAF CAL programs generated are simple but usable. Two illustrative runs by "students" are given in Appendix C for the program generated in Appendix A.

Subroutines are used in the STAF programs generated to manipulate counters, but are not available to the preprocessor user. Because of its basic nature, the preprocessor cannot produce sophisticated CAL programs. This inadequacy is further looked at in the next section.

6 DISCUSSION AND RECOMMENDATIONS FOR FURTHER WORK

Particular problems which have not been completely solved are now discussed with recommendations for their solution.

6.1 Structure of the CAL program

This most important problem involves the ability to define complex teaching strategies. I have now realised that it is wrong to assume that the teacher need not do much planning, as the production of sophisticated CAL programs involves substantial planning.

As it is good to separate the teaching strategy from the course content, the teaching strategy could be described first. Producing a CAL STAF program using the preprocessor would then take two steps. The teacher would first describe the teaching strategy before proceeding to the course content. The description of the teaching strategy is best done using directed graphs. The preprocessor would then use these graphs to prompt the teacher. Completing the CAL program with the course content is similar to the task performed by the present preprocessor.

A set of predefined teaching strategies can be stored as directed graphs for general use. The inexperienced teacher can write CAL programs using the basic module structure or one of these predefined strategies. The only foreseeable problem is attempting to illustrate the directed graphs clearly on the character oriented terminals.

6.2 Editing and Review

Review of the CAL program should not be a task of the preprocessor as the interpreter is more suitable. But it is necessary to be able to continue authoring the same program after a review.

Editing has not been implemented as it is of secondary importance. The preprocessor must be able to produce quality courseware

before such features are needed. Editing can be easily implemented on a modular basis. Modules of the CAL program can be added, deleted or moved around as their internal structure are complete and independent. To allow for editing, it is necessary to have intermediate code where frames have qualitative labels instead of specific labels. The CAL program is parsed to give the frames specific labels before it can be used.

This method of editing can be used to add or delete whole answer-matching schemes. Ordinary text editing (for example, the correction of spelling) is difficult to implement and could easily be performed by the CANDE editor.

6.3 Subroutines to manipulate Counters

To implement the use of counters, two problems will be met. They are:

i) How to identify a counter, and

ii) How to implement subroutine calls.

Counters are equivalent to numeric variables in other programming languages. These two problems are very similar to the problem of parsing arithmetic expressions. Implementation should not be difficult as this subject is well covered in the literature.

The teacher could be asked for the arithmetic expression. This is parsed to produce the necessary string of subroutines. Use of such subroutines will allow for the analysis of numeric student responses.

6.4 Specialised Answer-Matching Schemes

Specialised answer-matching schemes for YES/NO, TRUE/FALSE and multiple choice questions should be available to save the teacher

from such mundane details. This is easily implemented, yet can be very useful to the teacher.

7 CONCLUSIONS

The design of the preprocessor did not follow the design of other CAL authoring prompters because none were available. To meet the design criteria of this proposed preprocessor requires more time than was available.

The final structure discussed, though is not implemented, is important because the teaching strategy and the sophistication of the CAL programs depends on it. I consider this the most valuable contribution of the project. The preprocessor gives the skeleton for a more sophisticated preprocessor. Answer-matching and text display have been fully developed.

As a final point, Lower [4] says that modern authoring systems are not universal solutions, but rather, assists authors in particular categories of problems. This is true of the STAF system which is very suited for the objective and definite field of Science. The overall structure of the preprocessor could be used for other CAL languages which are more suited to other fields.

APPENDIX A

A SAMPLE PREPROCESSOR RUN

This appendix explains and points out the interesting points about the preprocessor through a sample run. The comments given are for the print-out on the facing page. The words the teacher types in are underlined in red. Notice that though the majority of the printout is by the preprocessor, no long paragraphs are printed by it.

Line 1 is a CANDE command to initiate a run by the preprocessor (which was then known as TALK4).

Lines 2, 3 and 4 are messages from CANDE.

Lines 5, 6 and 7 tells of how multiline input can be terminated. Multiline input is prompted by a "<?>" as seen in lines 23 to 27. Messages from the preprocessor have ">>" at the start of their lines to distinguish them from CANDE messages.

Lines 9 to 15 occurs once for each CAL program produced. It is to obtain a list of topics for the modules of CAL program to be generated. Lines 13 and 14 gives the two module titles for this CAL program.

The order of QUESTION (lines 31 to 33), MODEL ANSWER (lines 35 to 39) and ANSWER-MATCHES (line 41 onwards) helps the teacher to be consistent in his answers. Lines 41 to 52 shows the interaction necessary to produce an answer-match.

```
RUN TALK4
#RUNNING 6692
#6692 (COSC460LEE)CAL REMOVED ON CANDEPACK PK082 .
#?
>>  'STAF' PREPROCESSOR IS NOW READY AND RUNNING
>> ** END ALL MULTILINE INPUT WITH '#' IN THE FIRST COLUMN
>>      OF A NEW LINE.

>> PLEASE GIVE IN CHRONOLOGICAL ORDER THE MAIN TOPICS
>>  WHICH WILL CONSTITUTE THE STAF PACKAGE.
>>  NB*  KEEP EACH ENTRY TO A LINE AND END WITH A LINE-FEED.
>>          END THE LIST OF TOPICS WITH A '#' AS THE FIRST CHAR
  1.  IDENTIFYING A SKID
  2.  WHAT TO DO IN A SKID
  3.  #


**************************************************************************
>> NOW LET US LOOK AT TOPIC    1.
>> IDENTIFYING A SKID

>> DO YOU WANT TO ENTER TEXT BEFORE THE QUESTION? YES
<?>YOU ARE DRIVING ALONG ON A FROSTY WINTER'S MORNING.
<?>AS YOU ARE NEGOTIATING A BEND, YOU REALISE THAT THE CAR
<?>IS NOT RESPONDING TO THE STEERING WHEEL.
<?>
<?>#

>> ANY MORE TEXT? N

>> PLEASE GIVE QUESTION REQUIRING SHORT ANSWER
<?>DESCRIBE WHAT IS HAPPENING
<?>#

>> IS THERE A MODEL ANSWER TO THIS QUESTION? Y
>> PLEASE GIVE THE MODEL ANSWER
<?>YOUR CAR IS SKIDDING BECAUSE OF THE ICE ON THE ROAD.
<?>#
>> SHOULD THERE BE A PAUSE AFTER THIS MODEL ANSWER? N

>> GIVE A RIGHT ANSWER EXPECTED PLEASE.
<?>SKID
>> ARE EXTRA WORDS ALLOWED? YES
>> ARE EXTRA LETTERS ALLOWED WITHIN A WORD? Y

>> SKID
>> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? I
>> ANSWER WITH A 'Y' OR 'N' Y
>> GIVE ALL ALTERNATIVES, ONE TO A LINE.
>> MAXIMUM OF 5 LINES.
<?>SKD
<?>#

>> ANY MORE RIGHT ANSWERS? N
```

Lines 3 to 9 specifies the answer-match of the first wrong answer. For each module, there is at least one right and one wrong answer-match.

Lines 5 and 6 defines the strictness of match for the answer-match of the wrong answer. For the second wrong answer-match (lines 11 to 29) there is one extra component to the strictness of match (line 15) because the answer-match involves more than one word.

The dialogue in lines 18 to 29 clearly shows that the answer-match is on a word basis.

The final section (from line 32 onwards) defines the responses to be made in reply to the student's input. Replies are made for:

a) the Null Response (lines 33 to 36),

b) the Right Answer (lines 39 to 47),

and c) the two Wrong Answers (lines 48 to 58 and lines 1 to 7 on the next page).

A special response is used for the first wrong answer-match to give a more specific reply to the student.

```
1
2
3       >> WHAT IS THE TOTALLY WRONG ANSWER DO YOU EXPECT?
4       <?>PUNCTR
5       >> ARE EXTRA WORDS ALLOWED? Y
6       >> ARE EXTRA LETTERS ALLOWED WITHIN A WORD? Y
7
8       >> PUNCTR
9       >> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? NO
10
11      >> ANY MORE TOTALLY WRONG ANSWERS? YES
12      >> PLEASE GIVE THE RESPONSE EXPECTED
13      <?>STEER WHEEL BROKE
14      >> ARE EXTRA WORDS ALLOWED? Y
15      >> MUST THE ORDER OF OCCURRENCE OF WORDS BE KEPT? NO
16      >> ARE EXTRA LETTERS ALLOWED WITHIN A WORD? Y
17
18      >> STEER
19      >> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? N
20
21      >> WHEEL
22      >> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? N
23
24      >> BROKE
25      >> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? Y
26      >> GIVE ALL ALTERNATIVES, ONE TO A LINE.
27      >> MAXIMUM OF 5 LINES.
28      <?>BRK
29      <?>#
30
31      >> ANY MORE TOTALLY WRONG ANSWERS? N
32
33      >> CHOOSE ACTION TO TAKE WHEN STUDENT ENTERS A BLANK LINE
34                          0. GIVE MODEL ANSWER AND GO TO NEXT QUESTION
35                          1. ASK STUDENT TO TRY AGAIN
36      <?>0
37
38
39      @0 SKID$SKD !B02;
40      >> CHOOSE THE WANTED RESPONSE BY ITS NUMBER PLEASE
41                          0. NO RESPONSE
42                          1. SPECIAL RESPONSE
43                          2. GOOD ANSWER
44                          3. GOOD
45                          4. CORRECT
46                          5. YES
47      <?>4
48      %0 FUNCTR !B03;
49      >> CHOOSE THE WANTED RESPONSE BY ITS NUMBER PLEASE
50                          0. NO RESPONSE
51                          1. SPECIAL RESPONSE
52                          2. NO
53                          3. WELL ACTUALLY
54      <?>1
55      >> PLEASE ENTER YOUR OWN SPECIFIC RESPONSE
56      <?>YES , IT COULD BE A PUNCTURE BUT IS MORE LIKELY TO BE SKIDDIN(
57      <?>#
58      >> DO YOU WANT THE STUDENT TO HAVE ANOTHER TRY? NO
59
```

Lines 1 to 7 specifies the reply for the second wrong answer-match and completes Module #1. The "WELL ACTUALLY" is used as a connective to the model answer.

In Module #2, more advanced features is described. The topic is given in line 12 as a reminder for the teacher.

Two pages of text together with the question (lines 14 to 27) makes up the presentation of the problem for this module. Please refer to Appendix C to have a better idea of the effects.

If a pause is chosen (line 36), the student will need to press the Linefeed key to continue. A pause after the MODEL ANSWER is necessary if the model answer is long and the text for the next module is also long. Without a pause, the model answer can disappear at the top of a CRT terminal.

There are not many restrictions placed on input to the preprocessor. Leading blanks occuring where they should not (in line 55) are ignored.

```
%0 STEER WHEEL BROKE$BRK !B04;
>> CHOOSE THE WANTED RESPONSE BY ITS NUMBER PLEASE
                        0.  NO RESPONSE
                        1.  SPECIAL RESPONSE
                        2.  NO
                        3.  WELL ACTUALLY
<?>3


**********************************************************************
>> NOW LET US LOOK AT TOPIC   2.
>> WHAT TO DO IN A SKID

>> DO YOU WANT TO ENTER TEXT BEFORE THE QUESTION? YES
<?>YOU ARE SKIDDING TOWARDS A TREE.
<?>THERE ARE TWO THINGS YOU SHOULD DO.
<?>#

>> ANY MORE TEXT? Y
<?>ONE IS TO TAKE YOUR FOOT OFF THE ACCELERATOR.
<?>#

>> ANY MORE TEXT? NO

>> PLEASE GIVE QUESTION REQUIRING SHORT ANSWER
<?>WHAT IS THE OTHER THING TO DO?
<?>#

>> IS THERE A MODEL ANSWER TO THIS QUESTION? YES
>> PLEASE GIVE THE MODEL ANSWER
<?>YOU SHOULD STEER IN THE DIRECTION OF THE SKID.
<?>WHEN YOUR FRONT WHEELS GRIP THE ROAD, ONLY THEN SHOULD
<?>YOU CORRECT YOUR CAR'S DIRECTION.
<?>
<?>#
>> SHOULD THERE BE A PAUSE AFTER THIS MODEL ANSWER? NO

>> GIVE A RIGHT ANSWER EXPECTED PLEASE.
<?>STEER DIREC SKD
>> ARE EXTRA WORDS ALLOWED? Y
>> MUST THE ORDER OF OCCURRENCE OF WORDS BE KEPT? N
>> ARE EXTRA LETTERS ALLOWED WITHIN A WORD? Y

>> STEER
>> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? Y
>> GIVE ALL ALTERNATIVES, ONE TO A LINE.
>> MAXIMUM OF 5 LINES.
<?>TURN
<?>#

>> DIREC
>> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? Y
>> GIVE ALL ALTERNATIVES, ONE TO A LINE.
>> MAXIMUM OF 5 LINES.
<?>   TO
<?>FOR
<?>#

>> SKD
>> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? N
```

There are five anticipated wrong answers in this module. The first wrong answer-match in lines 10 to 16 checks for responses like "Step on the brake" and "Pull the hand break". Notice that the letters "BRK" will also match the wrongly spelt "break".

The second wrong answer-match (line 18 to 30) is for an incomplete answer. The student could have mentioned "STEER" or "TURN" but not give the direction to turn to.

Other wrong answers expected has to do with "CLUTCH" (lines 32 to 43), "JUMPING" (lines 45 to 52) and "LIGHT" or "GEAR" (from line 54 of this page to line 7 of the next page).

As can be seen, the answer-matching definition is a bit tedious. This could be shortened if the teacher knows how to construct his own answer-matches. The preprocessor could then be modified to accept complete answer-matches.

```
>> GIVE ALL ALTERNATIVES, ONE TO A LINE.
>> MAXIMUM OF 5 LINES.
<?>MOTION
<?>MOVE
<?>#

>> ANY MORE RIGHT ANSWERS? N

>> WHAT IS THE TOTALLY WRONG ANSWER DO YOU EXPECT?
<?>BRK
>> ARE EXTRA WORDS ALLOWED? Y
>> ARE EXTRA LETTERS ALLOWED WITHIN A WORD? Y

>> BRK
>> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? NO

>> ANY MORE TOTALLY WRONG ANSWERS? YES
>> PLEASE GIVE THE RESPONSE EXPECTED
<?>STEER
>> ARE EXTRA WORDS ALLOWED?
>> ANSWER WITH A 'Y' OR 'N' Y
>> ARE EXTRA LETTERS ALLOWED WITHIN A WORD? Y

>> STEER
>> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? YES
>> GIVE ALL ALTERNATIVES, ONE TO A LINE.
>> MAXIMUM OF 5 LINES.
<?>TURN
<?>#

>> ANY MORE TOTALLY WRONG ANSWERS? Y
>> PLEASE GIVE THE RESPONSE EXPECTED
<?>CLUTCH
>> ARE EXTRA WORDS ALLOWED? Y
>> ARE EXTRA LETTERS ALLOWED WITHIN A WORD? Y

>> CLUTCH
>> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? Y
>> GIVE ALL ALTERNATIVES, ONE TO A LINE.
>> MAXIMUM OF 5 LINES.
<?>CLCH
<?>#

>> ANY MORE TOTALLY WRONG ANSWERS? Y
>> PLEASE GIVE THE RESPONSE EXPECTED
<?>JMP
>> ARE EXTRA WORDS ALLOWED? Y
>> ARE EXTRA LETTERS ALLOWED WITHIN A WORD? Y

>> JMP
>> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? N

>> ANY MORE TOTALLY WRONG ANSWERS? Y
>> PLEASE GIVE THE RESPONSE EXPECTED
<?>LIGHT
>> ARE EXTRA WORDS ALLOWED? Y
>> ARE EXTRA LETTERS ALLOWED WITHIN A WORD? Y
```

Line 17 shows a reasonably complicated answer-matching scheme. A more descriptive sentence could be given after performing some processing on it. Instead, the actual answer-match is printed to show how valid matches are constructed. The teacher can learn from these examples and when the shorter alternatives to describe the answer match are implemented on the preprocessor, the teacher will be ready.

The robustness of the preprocessor is demonstrated in lines 25 to 27 and lines 34 to 36.

A simple help strategy is developed in lines 41 to 51, which gives a hint and allows the student to try again.

```
>> LIGHT
>> DO YOU HAVE ANY ALTERNATIVES FOR THIS WORD? Y
>> GIVE ALL ALTERNATIVES, ONE TO A LINE.
>> MAXIMUM OF 5 LINES.
<?>GEAR
<?>#

>> ANY MORE TOTALLY WRONG ANSWERS? N

>> CHOOSE ACTION TO TAKE WHEN STUDENT ENTERS A BLANK LINE
                    0. GIVE MODEL ANSWER AND GO TO NEXT QUESTION
                    1. ASK STUDENT TO TRY AGAIN
<?>1


 @0 STEER$TURN DIREC$TO$FOR SKD$MOTION$MOVE !B07;
>> CHOOSE THE WANTED RESPONSE BY ITS NUMBER PLEASE
                    0. NO RESPONSE
                    1. SPECIAL RESPONSE
                    2. GOOD ANSWER
                    3. GOOD
                    4. CORRECT
                    5. YES
<?>YES
>> PLEASE ENTER CHOICE AS A NUMERIC VALUE
<?>5
%0 BRK !B08;
>> CHOOSE THE WANTED RESPONSE BY ITS NUMBER PLEASE
                    0. NO RESPONSE
                    1. SPECIAL RESPONSE
                    2. NO
                    3. WELL ACTUALLY
<?>4
>> CHOICE GIVEN IS OUT OF RANGE. TRY AGAIN
<?>1
>> PLEASE ENTER YOUR OWN SPECIFIC RESPONSE
<?>BANG! YOU ARE NOW UP AGAINST THE TREE ***
<?>#
>> DO YOU WANT THE STUDENT TO HAVE ANOTHER TRY? N
%0 STEER$TURN !B09;
>> CHOOSE THE WANTED RESPONSE BY ITS NUMBER PLEASE
                    0. NO RESPONSE
                    1. SPECIAL RESPONSE
                    2. NO
                    3. WELL ACTUALLY
<?>1
>> PLEASE ENTER YOUR OWN SPECIFIC RESPONSE
<?>YES, YP\OU SHOULD REACT BY STEERING; BUT IN WHICK\H DIRECTION?
<?>#
>> DO YOU WANT THE STUDENT TO HAVE ANOTHER TRY? YES
%0 CLUTCH$CLCH !B10;
>> CHOOSE THE WANTED RESPONSE BY ITS NUMBER PLEASE
                    0. NO RESPONSE
                    1. SPECIAL RESPONSE
                    2. NO
                    3. WELL ACTUALLY
<?>3
```

Lines 1 to 11 produces code that allows the student to try again because his answer was quite out of point.

Finally, the preprocessor clearly informs the teacher of its completion.

Line 20 is a CANDE message.

```
%0 JMP !B11;
>> CHOOSE THE WANTED RESPONSE BY ITS NUMBER PLEASE
                    0. NO RESPONSE
                    1. SPECIAL RESPONSE
                    2. NO
                    3. WELL ACTUALLY
<?>1
>> PLEASE ENTER YOUR OWN SPECIFIC RESPONSE
<?>YOU WILL PROBABLY KILL YOURSELF....
<?>#
>> DO YOU WANT THE STUDENT TO HAVE ANOTHER TRY? Y
%0 LIGHT$GEAR !B12;
>> CHOOSE THE WANTED RESPONSE BY ITS NUMBER PLEASE
                    0. NO RESPONSE
                    1. SPECIAL RESPONSE
                    2. NO
                    3. WELL ACTUALLY
<?>2
>> WELL, YOU HAVE JUST FINISHED YOUR LAST TOPIC. GOODBYE **
#ET=28:56.2 PT=2.9 IO=0.7
```

APPENDIX B

THE STAF PROGRAM PRODUCED

This STAF program was produced by the preprocessor following the interaction given in Appendix A.

For those who want to understand the program lines 200 to 2400 is module 1, lines 2500 to 6200 is module 2 and the rest of the program are system responses.

```
LIST CAL :T
#FILE (COSC460LEE)CAL ON CANDEPACK
100 *0*
200 #Q01;0*
300 YOU ARE DRIVING ALONG ON A FROSTY WINTER'S MORNING.
400 AS YOU ARE NEGOTIATING A BEND, YOU REALISE THAT THE CAR
500 IS NOT RESPONDING TO THE STEERING WHEEL.
600
700 !$SR;Z00B01*
800 #B01;0*
900 DESCRIBE WHAT IS HAPPENING
1000 !$EVCX04;A01A01*
1100 #M01;0*
1200 YOUR CAR IS SKIDDING BECAUSE OF THE ICE ON THE ROAD.
1300 !Q02;
1400 #A01;0*
1500 @0 SKID$SKD !B02;
1600 %0 PUNCTR !B03;
1700 %0 STEER WHEEL BROKE$BRK !B04;
1800 !$LTCX01;N01$NR;M01$SR;Z01A01*
1900 #N01;0* !$SR;Z03M01*
2000 #B02;0*  !$SR;R04M01*
2100 #B03;0*
2200 YES ,  IT COULD BE A PUNCTURE BUT IS MORE LIKELY TO BE SKIDDING
2300 !M01;
2400 #B04;0*  !$SR;R07M01*
2500 #Q02;0*
2600 YOU ARE SKIDDING TOWARDS A TREE.
2700 THERE ARE TWO THINGS YOU SHOULD DO.
2800 !$SR;Z00B05*
2900 #B05;0*
3000 ONE IS TO TAKE YOUR FOOT OFF THE ACCELERATOR.
3100 !$SR;Z00B06*
3200 #B06;0*
3300 WHAT IS THE OTHER THING TO DO?
3400 !$EVCX04;A01A02*
3500 #M02;0*
3600 YOU SHOULD STEER IN THE DIRECTION OF THE SKID.
3700 WHEN YOUR FRONT WHEELS GRIP THE ROAD, ONLY THEN SHOULD
3800 YOU CORRECT YOUR CAR'S DIRECTION.
3900
4000 !Q03;
4100 #A02;0*
4200 @0 STEER$TURN DIREC$TO$FOR SKD$MOTION$MOVE !B07;
4300 %0 BRK !B08;
4400 %0 STEER$TURN !B09;
4500 %0 CLUTCH$CLCH !B10;
4600 %0 JMP !B11;
4700 %0 LIGHT$GEAR !B12;
4800 !$LTCX01;N02$NR;S02$SR;Z01A02*
4900 #N02;0* !$SR;Z03M02*
5000 #S02;0* !$SR;Z02A02*
5100 #B07;0*  !$SR;R05M02*
5200 #B08;0*
5300 BANG"! YOU ARE NOW UP AGAINST THE TREE ***
5400 !M02;
5500 #B09;0*
5600 YES, YOU SHOULD REACT BY STEERING, BUT IN WHICH DIRECTION?
5700 !$SR;Z04A02*
5800 #B10;0*  !$SR;R07M02*
5900 #B11;0*
6000 YOU WILL PROBABLY KILL YOURSELF ....
6100 !$SR;Z04A02*
6200 #B12;0*  !$SR;R06M02*
6300 #Q03;0*
6400  *** CHEERIO *** !$TM;*
6500 #R04;0*  CORRECT                          !$RS;*
6600 #R05;0*  YES                              !$RS;*
6700 #R06;0*  NO                               !$RS;*
6800 #R07;0*  WELL ACTUALLY                    !$RS;*
6900 #Z00;0*  PRESS LF TO CONTINUE  @0!A01;          !$RS;*
7000 #Z01;0*  DONT UNDERSTAND YOU PLEASE TRY AGAIN !$DCCX01; $RS;*
7100 #Z02;0*  GIVE IT A GO                          !$CLCXCX; $RS;*
7200 #Z03;0*  I THINK YOU HAVE HAD ENOUGH TRIES     !$RS;*
7300 #Z04;0*  HAVE ANOTHER TRY                      !$CLCXCX; $RS;*
7400 ##
#
```

APPENDIX C

TWO EXAMPLES OF "STUDENTS" RUN

Two sample runs by students are given to illustrate the STAF program generated.

This first example exemplifies the case when the student's responses were anticipated. The second example shows where the teacher has not fully covered all the alternatives. This shows that the phrasing of questions is very important.

Students responses are limited to one line by the STAF system. Students are prompted by a ">".

Lines 27 to 36 gives the example of a paged output.

```
RUN $SYSTEM/TUTOR
#RUNNING 6895
#?
 ENTER TEACHFILE NAME THEN PRESS LF KEY
TF/CAL
LOADING
THE PROGRAM IS NOW READY
REMEMBER TO TERMINATE ALL RESPONSES WITH LF
 DO YOU WISH TO USE A PREVIOUSLY SAVED FILE FOR THIS RUN?
ANSWER YES OR NO
NO
*** RUN AT 13:53 ON 100180

YOU ARE DRIVING ALONG ON A FROSTY WINTER'S MORNING.
AS YOU ARE NEGOTIATING A BEND, YOU REALISE THAT THE CAR
IS NOT RESPONDING TO THE STEERING WHEEL.

          PRESS LF TO CONTINUE
>_

DESCRIBE WHAT IS HAPPENING
>      I AM SKIDDING
          CORRECT

YOUR CAR IS SKIDDING BECAUSE OF THE ICE ON THE ROAD.

YOU ARE SKIDDING TOWARDS A TREE.
THERE ARE TWO THINGS YOU SHOULD DO.
          PRESS LF TO CONTINUE

>
ONE IS TO TAKE YOUR FOOT OFF THE ACCELERATOR
          PRESS LF TO CONTINUE
>

WHAT IS THE OTHER THING TO DO?
>      JUMP OUT OF THE CAR AS FAST AS POSSIBLE

YOU WILL PROBABLY KILL YOURSELF ....
          HAVE ANOTHER TRY
>      BRAKE

BANG ! YOU ARE NOW UP AGAINST THE TREE ***

YOU SHOULD STEER IN THE DIRECTION OF THE SKID.
WHEN YOUR FRONT WHEELS GRIP THE ROAD, ONLY THEN SHOULD
YOU CORRECT YOUR CAR'S DIRECTION.


 *** CHEERIO ***
END OF CAL
#ET=2:34.1 PT=1.6 IO=1.0
```

The two possible alternatives to Null Response are given lines 24 to 26 which gives the model answer and lines 42 and 43 which gives the student another try.

Line 23 gives the default response when the student's answer was not anticipated. In line 38, the student could have thought that his answer was not recognised because of the spelling error.

Although not shown, the program generated allows a maximum of 4 unanticipated responses and a maximum of 2 null responses.

```
RUN $ \SYSTEM/TUTOR
#RUNNING 6920
#?
 ENTER TEACHFILE NAME THEN PRESS LF KEY
TF/CAL
LOADING
THE PROGRAM IS NOW READY
REMEMBER TO TERMINATE ALL RESPONSES WITH LF
 DO YOU WISH TO USE A PREVIOUSLY SAVED FILE FOR THIS RUN?
ANSWER YES OR NO
NO
*** RUN AT 13:58 ON 100180

YOU ARE DRIVING ALONG ON A FROSTY WINTER'S MORNING.
AS YOU ARE NEGOTIATING A BEND, YOU REALISE THAT THE CAR
IS NOT RESPONDING TO THE STEERING WHEEL.

          PRESS LF TO CONTINUE
>

DESCRIBE WHAT IS HAPPENING
>      A STRONG WIND IS BLOWING YOU OFF THE ROAD
          DONT UNDERSTAND YOU PLEASE TRY AGAIN
>

YOUR CAR IS SKIDDING BECAUSE OF THE ICE ON THE ROAD.

YOU ARE SKIDDING TOWARDS A TREE.
THERE ARE TWO THINGS YOU SHOULD DO.
          PRESS LF TO CONTINUE
>

ONE IS TO TAKE YOUR FOOT OFF THE ACCELERATOR.
          PRESS LF TO CONTINUE
>

WHAT IS THE OTHER THING TO DO?
>      SWITCH OF THE ENGINE
          DONT UNDERSTAND YOU PLEASE TRY AGAIN
>      SWITCH OFF THE ENGINE@\
          DONT UNDERSTAND YOU PLEASE TRY AGAIN
>
          GIVE IT A GO
>      GO TO A LOWER GEAR
          NO

YOU SHOULD STEER IN THE DIRECTION OF THE SKID.
WHEN YOUR FRONT WHEELS GRIP THE ROAD, ONLY THEN SHOULD
YOU CORRECT YOUR CAR'S DIRECTION.


 *** CHEERIO ***
END OF CAL
#ET=3:21.5 PT=2.2 IO=1.0
```

APPENDIX D

REFERENCES

[1] STAF (A Computer Dialogue System for Teachers). User Guide for the B6700.

[2] ALPERT, D. 'THE PLATO IV SYSTEM IN USE: A PROGRESS REPORT' Proceedings of the IFIP Second World Conference on Computers in Education, page 181 -

[3] Dean, P.M. CAL authoring systems. Educational Technology 18, 4 (1978), pages 20-23.

[4] Lower, Stephen K. 'Authoring Languages and the Evolution of CAL'(Article)

[5] A brief introduction of directed graphs is given in AHO, HOPCROFT and ULLMAN. The Design and Analysis of Computer Algorithms, pages 50 - 52