Pre-publication draft of a paper which appeared in the *Proceedings of the Tenth Western Educational Computing Conference (WECC'85)* Western Periodicals Company, pages 151-157.

# ECCLES: AN ‘EXPERT SYSTEM’ FOR C.A.L.

**Thomas J. Richards and Geoffrey I. Webb**

*Department of Computer Science, La Trobe University, Victoria, Australia*

## Abstract

An authoring and lesson management system is described for Computer Assisted Learning in which lesson questioning and control flow arising from student response are generated at lesson time from an internal model of the lesson topic area. This approach permits the rapid authoring of conceptually complex lesson material. A languageless menu—driven authoring system rninimises the system familiarization time for lesson authors and enforces the construction of logically complete lessons. From the student’s point of view, the system appears as an expert system in the lesson subject area, with precise and detailed knowledge of the topic taught and of the causes of the student’s own errors.

## 1 QUESTION STRUCTURE AND TOPIC STRUCTURE

‘ECCLES’ stands for ‘Epistemically Complex Courseware Lesson Executive System’, which on the face of it is yet another authoring, lesson dispensing and student record management software package of the test-and-branch variety. Certainly in ECCLES a student gets asked questions about a chosen topic; and certainly his or her replies will affect what questions are asked next. But at that superficial level of student interface the similarity ends, for in ECCLES the student interaction sequence is generated by software that describes the *topic* not the *questions*. ECCLES can be thought of as an expert system that under stands the topic area it is going to teach, and uses that understanding to drive the student interaction.

This has some significant consequences, notably for the lesson author. In the normal type of test-and-branch CAL system, which we call *question-driven*, the flow of control through the teaching materials is determined by the interaction of the student’s answers with a complex schedule, usually tree-structured, of questions, sub-questions, and sub-sub-questions; all of which have to be thought up by the lesson author and specifically authored into the lesson file. Moreover, in many cases the author wishes to submit to the student tens of hundreds of ‘topic items’ for analysis, so must then write a tree of specific questions for each such item. In such cases it is easy to see why authors find the authoring to running time ratio is about a hundred to one. And it is in just such cases that ECCLES offers an alternative approach.

In ECCLES the author enters, just once, a description of the lesson topic area, plus a list of the topic items, and ECCLES does the rest. We have found that this reduces authoring time to something of the order of the running time. More significantly perhaps, it means that the devising of Complex and comprehensive lessons on complicated subject material is now a realistic activity.

We will make these remarks more concrete with an example.

## 2 A FRENCH VERBS LESSON

Consider the task of teaching a lesson on the French passé composé tense. Assuming a certain understanding of the topic by the student, you may wish to use a CAL program to test and reinforce a student’s ability to recognize and analyze sentences containing that tense. So, you arrange for the student to be presented with French sentences, some correct and others not. These are the *topic items* referred to above. You will want to ask the student a number of questions about each topic item: what is its auxiliary verb, is the sentence reflexive, with what does its subject agree, and so on. All the grammatical ingredients of correct use of the passé composé.

To do all that, you will need to create, for each topic item, a schedule of questions. Each question will require an associated specification of how to handle the students' possible responses and what to do with them - primarily remediation of wrong responses, and conditions for progress to further questions, until the full analysis of the current topic item French sentence is completed, or a student misunderstanding is diagnosed that makes further analysis of the topic item fruitless. Then the next French sentence can be presented, with its associated tree of questions.

It is this *question structured* approach to authoring a lesson that gives rise to *question driven flow of control* at lesson time. And it is this approach that makes it very hard to write an adequate lesson for a complex topic, due solely to the amount of authoring time that is required.

To write such a lesson using ECCLES, however, you devise one tree - structured analysis of the topic area. What types of auxiliary verb can there be? For each, what sort of verb-noun agreement do we have? In each type, how does the past participle get modified? The resulting tree gets compiled in ECCLES as the lesson driver code, resulting in what we call *topic structured lessons* with *topic driven flow of control* at lesson time. The only other job for the author is to write the list of topic items. At lesson time, when ECCLES presents a topic item to a student, it knows the correct analysis of that item in terms of the analysis of the topic area. Thus it can select and present the correct questions to the student about the topic item, and handle correctly the student's responses.

Question structured authoring is largely taken with issues of flow of control, that is, issues such as what should be done next if the student believes that the main verb in *La viande que tu as acheteé n'est pas très bonne* is reflexive? Such issues of flow of control are not of major concern during topic structured authoring, as the flow of control is determined by the logical structure of the topic under examination.

With topic structured authoring, the success of the lesson lies in the author's expert ability to provide a conceptual analysis of the topic to be taught, such as the passé composé. (In the case of the French lesson we are describing, the expert was Bob Hooke, Director of LaTrobe's Language Centre, who wrote the lesson for his students.) It is only where the topic to be taught has an analysable internal structure that topic structured authoring is of value; and the more complex the structure of that knowledge, the more ECCLES shines. That is why we named it as a system for Epistemically Complex Courses.

## 3 DECISION TREES

The problem implied by the above discussion is, how to arrange for a CAL lesson that will take the student through an analysis of epistemically complex topic items, where the constraints are:

1. The topic items are generically similar (e.g. all are sentences containing a passé composé construction)
2. The analysis required for each topic item will depend on the details of that topic item (e.g., the main verb is *revenir*).
3. The analysis required for each topic item will also depend on details of the student's understanding of that topic item.

Our answer is to use decision trees. A decision tree can be thought of as a tree to be traversed from root to one tip (a terminal node); and at each choice point (an internal node) in the tree a decision must be made somehow as to which branch to take leading out of that node. One way to make that decision is to tell the program which particular terminal node is to be reached. This identifies a unique path through the tree, so the program can work out which branch to take at each node.

To make a decision tree for CAL purposes, we decided to treat all internal nodes as being places where multiple choice questions are asked. These we call *question nodes*. Each branch out of a question node is associated with a possible answer to the question. These answers are known as *options*. Each branch leads to a further node. This further node can either be another question node, or a terminal node. The node that a branch leads to indicates the next stage in an analysis of a topic item for which the option associated with the branch is correct. If it is a question node, it indicates the next appropriate question to ask about such a topic item. If it is a terminal node, it indicates that there are no further questions that are relevant to such an item.

For each topic item, the program knows the correct terminal node, from which it can derive the complete path through the decision tree, and hence, the relevant questions for the topic item and the correct answers to those questions.

A lesson then consists of a series of exercises. Each exercise comprises of the presentation of a topic item to the student, and a detailed examination of the student's understanding of the item. The first question asked of the student is always the root node of the decision tree. As long as the student supplies the correct answer to questions, questioning always proceeds with the next appropriate question for the topic item under examination. This questioning continues until the student makes an error, or a terminal node of the decision tree is reached. This latter option indicates that all questions relevant to the current topic item have been presented to the student. As one exercise is completed, another topic item is chosen and a new exercise is commenced.

Figure 1 shows part of a decision tree for testing knowledge of the French passé composé. In it, question nodes are rectangles and terminal nodes are circles or ovals. Each terminal node represents one of the possible ways of individualising the test items. For example, the leftmost terminal node fixes the path to be followed by all grammatically correct sentences whose past participle, *avoir*, is followed by a direct object. Each question node rectangle contains the path descriptions of the paths leaving that node. The path descriptions at a node fix the logic of the question being asked at that node. Thus, at the left node on the second level down, the logic of the question is: determination of whether the main verb takes *avoir* or *être* as auxiliary, or is reflexive. Questioning about the topic item *Je l'ai frappé sur la bouche*, which takes *avoir* as auxiliary, must proceed along the left path out that node.

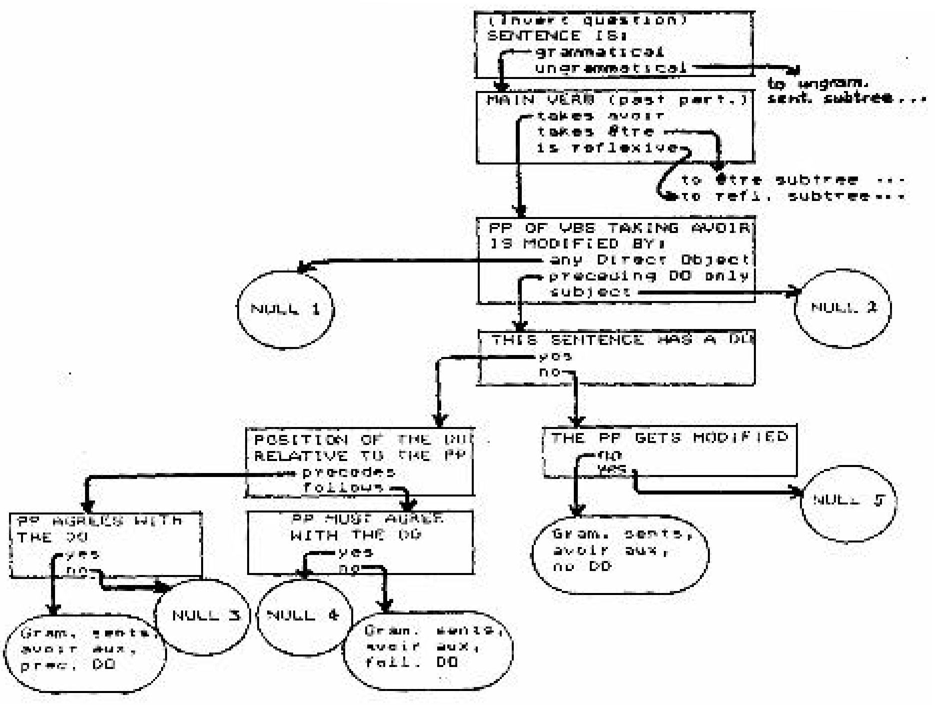

**Figure 1. Extract from a decision tree for testing understanding of the French passé composé tense.**

However, it is not sufficient to simply ask the student a question and then perform an internal analysis of their response. Rather, if the student identifies the correct answer to the question, this should be confirmed. Further, if they choose the wrong answer, they should be told so, and given an analysis of the mistake that they have made. The decision tree structure provides exactly the required environment in which to allow this feedback to be supplied. When presenting a question to the student for a particular topic item, ECCLES knows what the *correct answer* is and which answer the student has identified s being correct. It is only necessary then to associate with each option that is presented to the student, a matrix of *response texts*, each response text being associated with one of the options for the question. When the option that the matrix is associated with is correct for a topic item and the student selects the option that a text in the matrix is associated with, that text is displayed to the student. Figure 2 gives the response texts associated with the left most branch at the second level in Figure 1. So, when ECCLES presents to the student a sentence that takes *avoir*, such as our earlier example *Je l'ai frappé sur la bouche*, it presents the question at the MAIN VERB question node. The student then selects one of the three options that lead from that node. ECCLES then displays the reply to that choice which is shown in Figure 2. So if the student, mistakenly thinking that *je l'a* derives from the present tense of *être*, selects 'takes étre' then ECCLES will display the second message shown in Figure 2. ECCLES thus appears to the student to have a detailed understanding of both the topic item and of the precise causes of his or her own errors.

If the student answers at a question node by selecting a different option to the correct one for the current topic item, the system takes one of three possible *mismatch options* -

1. Further analysis of the current topic is stopped, a new topic item is selected and presented to the student, and questioning resumes at the root node.
2. The question is represented to the student; or
3. Questioning continues with the next node in the analysis for the current topic item.

Exactly which mismatch option is selected for each combination of correct answer and selected answer at a given question node is specified by the author when authoring the lesson.

| **SELECTED OPTION** | **RESPONSE TEXT** |
|---|---|
| takes avoir | "Yes, this sentence does take avoir." |
| takes etre | "No, 'a' is the present tense of avoir." |
| is reflexive | 'No, 'a' is not a reflexive pronoun or part of être." |

**Figure 2. Matrix of response texts for avoir.**

The term *correct option* is actually somewhat of a misnomer for the option that is identified to ECCLES as the one to follow for a given topic item. A difference between the so-called correct option and the option that a student selects for a topic item need not imply at all that the student has made a mistake. For instance, in the root node of the tree shown in Figure 1, the student is asked whether the test sentence is grammatical or not, as is required by the logic of the path descriptions. But, if they choose the response 'ungrammatical,' they are in fact choosing the left-hand path, not the right. And that is the path to be taken in questioning about grammatical sentences only. So a student will be taken down that path only if they mistakenly think that the sentence is ungrammatical. This is *inverted questioning* which is used to ensure that a student who gets the question right terminates at that point; and that only the erring student is probed by further questioning. Inverted questioning is appropriate for lessons whose objective is to provide practice on epistemically complex items after their nuts and bolts have been taught, and to provide remedial routines for mistakes. Ordinary questioning, on the other hand, is appropriate where the methods of analysis of epistemically complex material are still the primary object of tuition.

One theoretical matter remains. How does the program know which terminal node is the correct one for a given topic item? In our system, the answer is simple. Each topic item has associated with it the name or identification of its terminal node, and these associations are kept in the topic item file to be read in during the execution of the lesson. So, when a new topic item is presented to the student, the pro gram identifies its required path from its association.

Not all terminal nodes need be associated with actual or possible topic items. The terminal nodes in circles in Figure 1 represent necessarily incorrect choices and so can be reached by no test sentence (ungrammatical ones go to a sub-tree not shown in Figure 1 .) These null nodes are a side- effect of the logic of a decision tree.

## 4 THE AUTHOR INTERFACE

The task of authoring a decision tree requires a good deal of clerical overhead, to keep track of where you are in the tree, what question node is currently being authored for, and the like. How ever, it is not hard to design the authoring pro gram to take care of all such details. The only such clerical work that the author needs to do is to work out beforehand (preferably on a very big sheet of paper) the architecture of the tree he or she wants, just as in Figure 1. That, plus any decisions about inverted questioning, is all that the author need bring to the terminal for an authoring session.

Then the authoring program, by the use of such devices as menus, windows of relevant information, and exploitation of the screen characteristics and cursor control offered by smart terminals, guides the author in quite a foolproof way through each step of lesson creation. At each stage during the authoring process the screen looks in all relevant ways exactly how it will appear to the student at that step during lesson presentation, except of course that the topic item is necessarily absent.

A notable feature of authoring with ECCLES is that except for answering occasional questions (by single letter menu selection), the author need type nothing except the actual text that will appear to the student. There is no authoring language to learn at all.

There is one further task for the author - to write the file of topic items, each with their associated terminal node reference. This is done using an internal editor which is menu-driven in the same way as the decision tree authoring pro gram, and which provides a similar screen interface.

There is one problem about a CAL system that generates its questions purely from knowledge of the topic and not from specific topic items - it cannot make reference to aspects of any particular topic item. It could not, for instance, tell the student that in the topic item sentence *Je l'ai frappé sur la bouche*, the past participle is *frappé* or that the auxiliary verb is *ai*. Note however it could tell the student that the past participle in this sentence is masculine, for that is nonspecific information easily generated from the topic knowledge encoded in the decision tree. The first response text in Figure 2 illustrates this sort of nonspecific reply.

To overcome this limitation about specific references, ECCLES provides the lesson author with a straightforward boilerplate macro facility, akin to the ability of a word processor to insert specific names into form letters. These macros can be inserted into any material in the decision tree which will print onto the screen (e.g., a response 'The past participle here is *past-part* where *past-part* is a macro.) Then, for each topic item specified, the correct value for each macro is also specified and stored with it in the file of topic items. The desired substitutions are then made by ECCLES at lesson presentation time. Thus the problem is entirely overcome in a simple manner. 0 way of illustration, the second and third response texts in Figure 2 are specific and appear there just as a student would read them. We made them by writing the response texts as "No, '*aux*' is the present tense of *avoir*" and "No, '*aux*' is not a reflexive pronoun or part of *être*." - where *aux* is a macro which expands to the auxiliary for the topic item under examination. Now, for each topic item that we enter for the French passé composé lesson, ECCLES asks us for the corresponding value for *aux*

## 5 USING ECCLES

ECCLES version 1.0, written in Franz Lisp, is available for transportation to educational institutions for a nominal licensing fee. It will run on VAXes under VMS, and it will run under

Unix. A smaller version written in C is currently under development, which will make ECCLES available on personal computers.

Two earlier papers, (1) and (2), describe the system and its behaviour; and a reference manual (3) and an installation and management manual (14) are available for authors.

## References

[1] Richards, Thomas J., Hooke, Robert, & Webb, Geoffrey I., “Automatic Authoring of Complex and Analytical Question-Answer Lessons,” Proceedings of the First Conference on Computer- Assisted Learning in Tertiary Education 1983, Brisbane, University of Queensland, pp.282-293.

[2] Richards, Thomas J., Webb, Geoffrey I., & Bodnar, Steven, “The ECCLES Intelligent Computer-Assisted Learning System,” Proceedings of the Second Conference on Computer-Assisted Learning in Tertiary Education 1984, Brisbane, University of Queensland, pp.232-235.

[3] Webb, Geoffrey I., “The ECCLES Version 1.1 Reference Manual”, La Trobe University Department of Computer Science, Tech. Rep. 1985.

[4] Webb, Geoffrey I., “The ECCLES Version 1.1 VAX VMS Installation and Management Manual”, La Trobe University, Department of Computer Science, Tech. Rep. 8, 1985

## Biography

Richards, a philosopher and logician by training, is currently a Senior Lecturer in Computer Science at La Trobe University, where he specializes in Artificial Intelligence, in particular automated reasoning. His D.Phil, from Oxford University, was on formal reasoning about belief systems.

Webb, a philosophy and computer science graduate, is a PhD student in Computer Science at La Trobe University, where he is concurrently taking a Diploma in Linguistics. His PhD is on Knowledge-Based Computer-Aided Learning. His main research interests lie in the fields of’ CAL and Artificial Intelligence - particularly knowledge-representation and natural language processing.

*This research has been supported by grants from the Schools of Humanities and Physical Sciences at La Trobe University