Pre-publication draft of a paper which appeared in the *Proceedings of the Fifth Annual Conference of the Australian Society for Computers in Learning in Tertiary Education (ASCILITE-87)* pp 257-285

Generative CAL and courseware abstraction

Geoffrey I. Webb

Computing and Information Studies, Griffith University

Abstract

Courseware abstraction is an approach to CAL whereby the lesson author creates a general parameterised CAL lesson that is then applied to many concrete examples. This approach has the following advantages:

- it provides a powerful framework within which to adapt tuition to a student's knowledge and aptitude;
- it encourages the development of detailed treatments of the subject matter;
- it reduces the cost of lesson development as a ratio to student lesson time; and
- it enables large numbers of examples to be made available for individual students.

Generative CAL is an example of courseware abstraction. It is argued that the advantages of generative CAL do not arise directly from the generation of the examples to be examined but rather can be directly attributed to the use of courseware abstraction.

INTRODUCTION

Generative computer-aided learning is a CAL technique that was developed in the late sixties (Uttal, Pasich, Rogers & Hieronymous, 1969; Uhr, 1969). A generative CAL system generates unique problems and presents these to the student for solution. The system must be able to evaluate and critique the student's solution to the problem. This implies that the problems to be generated belong to a class of problems for which the system has the capacity to generate, evaluate and critique novel in stances.

Most generative CAL systems have been in the domain of mathematics. This is not a matter of coincidence. Mathematics is one of the few domains for which it is relatively simple for the computer to be provided with the capacity to generate, solve and critique students' solutions to novel problems. Once one leaves the domain of mathematics, relatively difficult AT based techniques are required to create generative CAL. (See, for example, Sleeman and Brown, 1982).

BENEFITS OF GENERATIVE CAL

Generative CAL offers several benefits:

- By selecting problems with appropriate attributes it is readily possible to adapt to variations in student's initial competence and learning rate within a do main.
- Because one general treatment of a domain is defined that is then applied to many instances from that domain, the lesson author is encouraged to make the treatment more detailed than if a separate treatment has to be written for each example to be examined as is the case with normal programmed learning.
- Because only the one general treatment has to be written, if the system is able to generate many examples from the domain and consequently the one general treatment is presented

to the student a large number of times then the relative cost in authoring time to student terminal time is likely to be very low. This can make the approach very cost effective.

- The student is provided with the opportunity to examine as many problems as may prove desirable, all within the one consistent framework.

However, a careful examination of these benefits will show that none of them result directly from the fact that the computer generates the problems to be examined. Rather, they result from the fact that one general treatment of a domain is applied to many specific instances from that domain. In short, the benefits of generative CAL all arise from the fact that it is a technique that employs an approach to creating CAL that will be referred to as *courseware abstraction*.

Courseware abstraction is a method of creating computer-based courseware that is both time saving for the lesson author and encourages thorough treatment of the subject matter. The author creates once only a detailed abstract CAL treatment of the subject matter. This treatment is parameterised. That is, those aspects of the subject matter that change from example to example are identified, and a variable is provided for each. This parameterised abstract treatment is then applied to a body of specific examples from the subject area. Associated with each example are the correct values for that example for each parameter in the abstract treatment. The re suit is a detailed treatment of the particular example for the student. The one lesson can then be presented to the student many times, each time using a different example and providing a relevant analysis for that example.

Generative CAL is one approach to CAL that employs courseware abstraction. The vast majority of generative CAL systems have been for mathematics. As stated above, this is not a matter of coincidence.

Generative CAL systems must meet the following requirements:

- They must be able to generate meaningful problems.
- They must be able to solve the problems generated.
- They must be able to monitor, remediate and assist the students' attempts to solve the problems generated.

As mentioned above, although this is reasonably tractable for arithmetic domains, once one leaves the field of mathematics very sophisticated artificial intelligence based systems are required. The development and operating overheads of such systems are prohibitive for most CAL applications.

However, courseware abstraction does not require that the system should generate the problems that the student is to examine. Rather, the course author can create a body of examples that the system can then use, thus avoiding the problem of generating novel problems. This approach has been used with success by a number of systems. It is a key factor in the power of the ECCLES (Richards & Webb, 1985) and DABIS (Webb, 1986) systems. Most AI based CAL systems utilise it. Despite being a key factor in many systems, this approach to CAL has not previously been identified and the role that it plays has not previously been recognised.

As the generative approach has been well documented, the rest of this paper will address non-generative approaches to courseware abstraction. To provide a con text for this discussion the next section examines a typical example of an application for courseware abstraction - CAL treatments of the French passé composé. The remaining sections provide a general discussion of the advantages and applicability of the approach and means of implementing it in existing CAL languages.

A COMPUTER-BASED LESSON ON THE FRENCH PASSÉ COMPOSÉ

The passé composé - the past tense verb agreement system - is an extremely difficult aspect of French grammar. The past tense is marked by the presence in a phrase of one of the auxiliary verbs *être* and *avoir*. Each verb can normally only appear with one of these auxiliary verbs. This aspect of French grammar has many irregularities. That is, for many French verbs it is not obvious which auxiliary verb the main verb takes. This is simply a matter of acquired knowledge.

The agreement rules of the passé composé can be summarised as follows:

- If there is a reflexive pronoun in the phrase as either the direct or indirect subject of the main verb then -
    - If the reflexive pronoun is the direct object of the main verb then the verb and direct object must agree in both number and gender. If the reflexive pronoun is the indirect object of a pronominal main verb then the verb must usually be unmarked for both number and gender. This means that it takes the masculine singular form. (The reflexive pronoun is treated here as the direct object of the verb in phrases such as *Elle s'est lavée* but not in phrases such as *Elle s'est lavé les mains*.)
    - Irrespective of which auxiliary verb the main verb normally takes, the phrase must contain the auxiliary verb *être.*
    - If there is no reflexive pronoun in a phrase and the main verb takes the auxiliary verb *être* then it must always agree with the subject of the phrase in both number and gender.
    - If a main verb takes the auxiliary verb *avoir* and there is a direct object in the phrase that precedes the main verb then the main verb must agree with it in both number and gender. If there is no direct object, or if the direct object follows the verb then the main verb must be unmarked for both number and gender.

To further complicate matters, the auxiliary verbs *être* and *avoir* take many different forms depending upon their context, making it difficult for the student to identify whether they are present.

From the complexity of the above description it should come as no surprise that students of French grammar typically experience great difficulty in mastering this aspect of the grammar. It is clearly of benefit to students to provide them with practical experience in identifying whether particular French phrases do conform to these rules and, if not, how exactly they conflict. In short, this aspect of French grammar is a prime target for CAL.

The only practical manner in which to provide this practice is to create a large series of exercises, $E_1, E_2, ...E_n$, each of which examines a separate French phrase, P_i such that E_i examines P_i.

The standard test and branch approach to the authoring of such a lesson is to create a list of example phrases to be examined and for each of these examples write a CAL segment that provides a detailed exercise examining that example. This will be called the *traditional approach*. This approach can be characterised as the creation of CAL segments $s_1, s_2, ...s_n$, such that s_i generates the interactions with the student required by E_i. As a result, the provision of *n* exercises will require the creation of *n* CAL segments.

It would be necessary in such a lesson to examine the following questions, among others, with relation to the examples:

- Is the phrase grammatical?
- What is its main verb?
- What is its auxiliary verb?
- Does the main verb agree with its subject?
- Does the main verb agree with its direct object?
- What is the main verb's number?
- What is the main verb's gender?
- Does the main verb have a direct or indirect object?
- Does the direct object precede or follow the main verb?
- Does the phrase contain a reflexive pronoun?

However, not all of these questions would apply to all of the examples. For example, it would only be important to determine the relative location of the direct object and main verb of a phrase if the main verb took *avoir*.

Each exercise would have to examine only those questions that are relevant to the example on which it is based. Each relevant question would be asked of the student. The student's answers would be evaluated and appropriate feedback provided.

The structure that each exercise might take would be to first enquire of the student whether s/he believes the example to be grammatical. If the student answers correctly then it can be assumed that s/he can correctly analyse the example and another example can be selected and examined. If this assumption is incorrect and s/he has only guessed the correct answer, this would be picked up at a later stage when s/ he again has to analyse a similar phrase.

If the student believes that the example is ungrammatical and it is not, each of the reasons why it could be ungrammatical should be examined and the student should be shown how they do not apply to the example. Conversely, if the student believes that the phrase is grammatical and it is ungrammatical, then s/he should be shown exactly how it is ungrammatical. Figure 1 shows a fragment of such an example exercise written in Common Pilot. This fragment tests if the student can identify the direct object in the phrase *Jeanne je l'ai vue hier*. Common PILOT is described in Khieraty & Gerhold (1980).

```
R:     Examine whether there is a direct object in the phrase.
T:     Is there a direct object in "Jeanne je l'ai vue hier"?
U:     ANSYES
TY:    Yes, well done!
TN:    Wrong, "L'" is the direct object in this phrase.
JN:    @P
T:
T:     What is the direct object in "Jeanne je l'ai vue hier"?
A:
M:     %L'%
TI:    Very good!
JY:    AGREE
M:     %JE%
TY:    No, "JE" is the subject of this phrase.
TY:    The direct object is "L'"!
TN:    No, the direct object in this phrase is "L'"

*AGREE
```

ANSYS is a subroutine that sets the Y conditioner if the student gives an affirmative response and sets the N conditioner for a negative response.

FIGURE 1: A fragment of a standard test and branch lesson on the French passé composé

Let us assume that we are going to provide one hundred exercises for the student. This is far fewer than would be desirable, but it will do to demonstrate the point.

For each exercise there will be a complex schedule of:

- frames of text to be displayed to the student;
- questions to be asked of the student;
- answer evaluation procedures to apply to the student's responses; and
- conditional branching dependent upon the student's responses.

There will be some similarities in this schedule from exercise to exercise, in that the text frames, questions and answer evaluation procedures will be drawn from one pool, but the exact set used will vary greatly from exercise to exercise.

If the traditional approach were used each of these exercises would have to be authored as separate CAL segments. Certainly, there would be some savings in authoring time - due to being able to copy frames, answer evaluation procedures, etc from one exercise to the next - but each exercise would still have to be separately manually constructed by the author.

It is probable that each CAL segment would take at least 100 minutes to author. Assuming that each exercise would occupy approximately 1 minute of the student's time, as each exercise is embodied in a separate CAL segment, this results in an authoring ratio of 100:1. This estimate is probably quite conservative. A ratio of 200 authoring hours to one student hour is not uncommon for this authoring methodology.

On these figures, for 100 exercises the author could expect to spend 10,000 minutes, or approximately 165 hours. All of this for 1 hour and 40 minutes of student lesson time! This amount of time might be feasible for a software house, but it certainly is not for a teacher wanting to provide some remedial exercises for a few students!

This would be a prime target for generative CAL if only it were possible for the computer to generate novel examples and to determine the correct analyses for those examples. The creation of such a system may be currently feasible using state of the art artificial intelligence programming. However, the developmental cost involved would almost certainly far exceed that of the traditional approach.

The alternative approach that is being proposed is to write once only some generalised courseware which can then be used to examine any of a large number of concrete examples from the domain in question. This can be described as the creation of a single CAL segment *S(i)*, which takes as a parameter a description of the item to be examined, and generates the appropriate exercise for the student, E_i. In the case of the French passé composé this means writing a lesson that can examine any combination of verb type and agreement (or lack of agreement) for any phrase. Such a lesson has all the advantages of a generative system with the added bonus that it is feasible to implement!

Figure 2 contains a portion of an example lesson which utilises this approach, again written in Common Pilot. The code in Figure 2 is a subroutine which examines whether the student can identify the direct object in a phrase. This lesson fragment is functionally identical to that in Figure 1. That is, given that the fragment in Figure 2 is being applied to the phrase examined in Figure 1, both respond identically to the student. Despite the fact that the abstracted code can be used in examining any phrase, it is less than twice the size of the traditional code which can only be used to examine one particular phrase. The greater utility of the abstracted code is gained at very little cost.

Clearly the abstracted courseware requires some means of determining the details of the concrete example under consideration. Otherwise it cannot determine what the correct answers are for any particular ex ample phrase. The lesson from which the fragment in Figure 2 is taken reads the phrase and various parameters about it in from a file. Figure 3 shows the parameters that are read in for exercises on the phrases *Jeanne je l'ai vue hier* and *Nous nous sommes arrêtés devant la bibliothèque*.

Using this authoring method, if we assume that it will take the author approximately 40 hours to specify the initial lesson and then approximately 1 minute to specify each example phrase and its parameters, then we have an authoring time of 2500 minutes for the lesson with 100 examples. This is an authoring ratio of 25:1, a fourfold improvement over the traditional approach. It is not really important whether the (probably already excessive) figure of 40 hours for the specification of the abstracted lesson is not accepted. No matter how exorbitantly this figure is inflated, the ratio will still come out in favour of course- ware abstraction in the long run. This is because, having written the abstracted les son and being in the position that each new phrase only takes a minute to specify, the author is unlikely to stop at one hundred examples. The more examples that are specified, the better the authoring ratio will be. This contrasts sharply with the traditional approach where the authoring ratio remains constant no matter how many examples are provided.

```
*EXAMDO

R:
R:    Examine whether there is a direct object in the phrase.
T:    Is there a direct object in “$A$“?
U:    ANSYES
J(D$=“”): NODO
*DO
TY:   Yes, well done!
TN:   Wrong, “$D$ “ is the direct object in this phrase.
EN:   NEXTPH
T:    What is the direct object in “$A$”?
A:
T(%B = D$): Very good!
EC:
T(%B = E$ & E$ <> “”): No, “$E$ “ is the indirect object.
TC:   “$D$ “ is the direct object!
EC:
T(%B = C$): No, “$C$ “ is the subject of this phrase.
TC:   The direct object is “$D$“!
EC:
T:    No, the direct object in this phrase is “$D$
E:
*NODO
TN:   Excellent!
TY:   Wrong. There is no direct object for this phrase.
EY:   NEXTPH
E:
```

ANSYES is a subroutine that sets the Y conditioner if the student gives an affirmative response and sets the N conditioner for a negative response.

C is the subject of the phrase.
D is the direct object of the phrase.
E is the indirect object of the phrase.

FIGURE 2: A fragment of an abstracted lesson on the French passé compose

Several points should be noted about the lessons of which portions appear in Figures I and 2. First, as already mentioned, for the phrases covered by the traditional lesson, both lessons are functionally identical. The traditional lesson only covers three phrases. However, with just these three phrases the author has already written less code (albeit more complex) for the abstracted lesson than for the traditional one. Further, the smaller abstracted lesson can already handle far more than just the three examples that the traditional lesson handles. Indeed, the abstracted lesson can handle any grammatical phrase with a reflexive pronoun as the direct object or with a main verb that takes *avoir*. All that is now needed to make the abstracted lesson complete are short treatments of phrases with reflexive pronouns as indirect objects and of verbs that take *être* and a treatment of ungrammatical sentences. The latter would be approximately the same size again as the existing lesson code. By comparison, the task of the author using the traditional methodology has barely begun with the three phrase example which has been written.

The final point that should be made about the example lessons is that it is not important whether the reader believes that they provide a good or a bad treatment of the subject matter. They are provided only to serve as a concrete example of the difference between abstracted and non-abstracted courseware. Courseware abstraction is not tied to this programmed learning based style of lesson. For example, the GREATERP system involves the application of a general lesson to a sequence of concrete examples. Thus, it is also an abstracted lesson. However, it clearly does not use a programmed learning based approach to CAL. Rather, it utilises extremely sophisticated AI based CAL.

Phrase	*Jeanne je l'ai*	*Nous nous sommes arrêtés vue hier devant Ia bibliothique*
Grammaticality	GRAMMATICAL	GRAMMATICAL
Subject	*je*	*nous*
Direct Object	*l'*	*nous*
Indirect Object		*la*
bibliothèque		
Reflexive Pronoun		*nous*
Verb	*vu*	*arrêtis*
Auxiliary Verb	*avoir*	*être*
Location of direct object	PRECEDING	PRECEDING
Number	SINGULAR	PLURAL
Gender	FEMININE	MASCULINE

FIGURE 3: Parameters for two examples from the abstracted passé compose

THE ADVANTAGES OF COURSEWARE ABSTRACTION

An improved ratio of authoring to student time is just one of the advantages of courseware abstraction.

Lesson validation is far easier with abstracted courseware. With the traditional approach, the author will never be able to thoroughly determine that no errors lie hidden in some section of code in some exercise or other. With courseware abstraction, any error in the abstracted courseware will quickly come to light, as the same code is used for every exercise. Once the generalised lesson has been debugged, the entire course can easily be validated simply by checking that the parameters passed into the system are correct. If the parameters have been given sensible names then this is straight forward. The author need only scan through a list like that in Figure 3 in which any errors will be transparently obvious.

Another advantage of the methodology is that it encourages the author to provide an exhaustive treatment of the topic. With the traditional approach the author knows that for every aspect that is built into the lesson there is going to have to be separate code written for each example to which it applies. This can result in the author having to choose between providing a thorough treatment of a smaller than desirable set of examples, or a less thorough than desirable treatment of a larger body of examples. This is not true with abstracted courseware authoring because the author only has to provide the one thorough treatment of the domain. This is then applied to all examples examined thus ensuring that every example receives the same level of analysis.

Abstracted courseware is also easier to update. If the author decides that some aspect of the original treatment of the domain under examination should be changed, such a change need only occur in one place in the generalised lesson. Under the traditional approach, any such change would need to be made separately for each exercise to which it applied. This would cause two problems:

a. determining which exercises the change applies to; and
b. changing the lesson code in each exercise.

Another advantage of courseware abstraction is that it provides a framework in which it is readily possible to adapt instruction to each student's current level of expertise in a domain. If the CAL system is able to identify relevant features of the examples that it is examining, such as their relative difficulty, and is able to judge from the student's performance how well they perform in relation to these features, then it is relatively straight forward to select examples for examination

that are appropriate for the student's current needs. For instance, if the examples are graded with regard to difficulty, and the system can determine that the student makes no errors when examining examples at difficulty level one, makes errors for 50% of the examples at difficulty level two and 90% of the examples at difficulty level three then it will be able to determine that it should concentrate on examples at difficulty level two. All that it need do is start at the easiest difficulty level and work its way down to the more difficult levels, remaining at each difficulty level until the student achieves a set success rate.

But difficulty level is not the only or the most powerful distinguishing feature that could be taken into account. Consider the abstracted passé composé lesson. If the student is not making errors for examples that take the auxiliary verb *avoir* but is for examples that take the auxiliary verb *être*, then it will be quite trivial for the system to identify this fact and to concentrate on the examples for which the student needs further tuition—those which take the auxiliary verb *être*. All of the information that it needs in order to determine this - that is, the auxiliary verb for each example - is already explicitly stated and thus readily accessible.

A final advantage of abstracted courseware is that it greatly facilitates the analysis of a student's performance during the lesson. If a simple record is maintained of the questions at which the student makes mistakes, then it will be readily possible to determine what difficulties the student is experiencing. For instance, if the student often specifies the wrong word as being the main verb in the phrase then it is clear that they are having difficulty identifying the main verb. The standard test and branch approach makes this information far more difficult to obtain as there is no clearly defined structure by which to assign identity to similar questions in different exercises. At best, this becomes another chore for the author to look after. At worst, this information is simply not collected.

THE APPLICABILITY OF COURSEWARE ABSTRACTION

Courseware abstraction is clearly not a methodology that can advantageously be applied to all domains for which a computer-based lesson may be written. Rather, courseware abstraction only comes into its own when large numbers of concrete examples are to be examined in terms of a general theoretical framework.

An obvious area in which this applies is grammar. Typically languages have very complex grammatical systems for which it is advantageous for the student to examine large numbers of specific examples within a general framework. The passé composé lesson is an example of just one such lesson.

However, languages are far from the only domains to which such lessons may be applied. A lesson has been developed under the DABIS system (Webb, 1986) that examines the leading theories in the mind/body debate within philosophy. The lesson identifies a number of central questions within this debate and examines how each key position in the debate has a different set of answers to these questions.

Another possible use for the methodology is in the control of practical classes where the students must follow a strict order of enquiry to reach some conclusion. For instance, the students could be given a chemical solution and then the program could guide them through the step by step process of identifying its chemical composition. The same lesson could manage the examination of any of a large number of chemical solutions.

Courseware abstraction is a methodology of wide but not unlimited application.

COURSEWARE ABSTRACTION IN EXISTING CAL SYSTEMS

As is demonstrated with the Common Pilot lesson from which the fragment in Figure 2 is taken, courseware abstraction is possible using at least some existing CAL languages.

All that is required for a CAL authoring system to allow courseware abstraction is for it to support some method of specifying variables and then assigning them different values for different examples; and for it to allow conditional branching dependent upon the value of those variables.

Probably the simplest way to implement this in most existing CAL languages would be the approach taken in the lesson fragment in Figure 2. That is, to define a set of variables which are then read in from a file. For each example that is presented a new set of values is read into the variables from the file. Once a lesson has been written, a text editor can then be used to create and update the input file.

However, for courseware abstraction to be possible, it is not necessary for a CAL language to support files. If the language supports variables then it is possible sim ply to assign values to them within the program. Thus, each time that an exercise begins a new set of values are assigned to the variables.

Although courseware abstraction is possible using many existing CAL languages, specialised systems, such as ECCLES (Richards & Webb, 1985) and DABIS (Webb, 1986), that take advantage of the special features of the methodology increase its benefits to the author.

CONCLUSION

Courseware abstraction provides a methodology for the authoring of computer-based courseware which can create detailed lessons of a type not feasible through other approaches due to their huge cost in authoring time and difficulties of maintenance.

The benefits that courseware abstraction has to offer have already been demonstrated by the success of generative CAL. Among the attractive features of courseware abstraction are reduced authoring time; the provision of a convenient frame work for adapting tuition to the student's level of expertise; improved lesson verification; improved lesson modifiability; and improved student analysis.

Although courseware abstraction has been used in many previous CAL systems, there does not appear to have been any appreciation of the fact that it was courseware abstraction to which many of the desirable features of these systems could be attributed. This is particularly notable in the case of generative CAL where the benefits of the approach have been attributed to the generation of the problems to be examined whereas they can actually be attributed to the use of courseware abstraction.

It is to be hoped that a better understanding of factors underlying the success of these approaches will in turn lead to the development of yet better approaches.

Acknowledgements

The research presented in this paper grew out of the ECCLES project which was founded and led by Tom Richards. Thus, the ideas presented herein have directly evolved from Tom's original inspiration for ECCLES. I am also deeply indebted to Bob Hooke and Roly Sussex who advised me on the finer points of French grammar during the preparation of this paper. The lesson fragments in Figures 1 and 2 are based on a lesson written for the ECCLES system by Bob Hooke.

References

[1] Khieriaty, L. & Gerhold, G. (1980). *COMMON PILOT LanguageReferenceManual.* Bellington WA: Western Washington University.

[2] Richards, T. J. & Webb, G. I. (1985). **ECCLES: an "Expert System" for CAL**. In *Proceedings of the 1985 Western Educational Computing Conference*, Oakland, C.A., pp. 151-157

[3] Sleeman, D. & Brown, J. S. (Eds). (1982). *Intelligent Tutoring Systems*. London: Academic Press.

[4] Uhr, L. (1969). **Teaching machine programs that generate problems as a function of interaction with students**. In *Proceedings of the 24th National ACM Conference*, pp. 125-134.

[5] Uttal, W. R., Pasich, T., Rogers, M. & Hieronymus, R. (1969). **Generative Computer Assisted Instruction**. *Communication* 243, Mental Health Research Institute, University of Michigan.

[6] Webb, G. I. (1986). **Knowledge Representation in Computer-Aided Learning: The Theory and Practice of Knowledge-Based Student Evaluation and Flow of Control**. *PhD thesis*, LaTrobe University, School of Mathematical and Information Sciences.