Pre-publication draft of a paper which appeared in the Proceedings of the First Australian Artificial Intelligence Congress (1AAIC'86), Section B pp 1-7.

Knowledge-Based Flow of Control in Computer-Aided Learning

Geoffrey I. Webb

Department of Computer Science, La Trobe University, Bundoora, 3083

Abstract

In this paper I examine the utilisation of knowledge representation in Computer-Aided Learning (CAL) with the aim of establishing knowledge-based CAL techniques that are best suited to current technology. Most existing knowledge-based CAL systems attempt to generate the entire instructional sequence directly from a domain knowledge base. Such systems suffer from several limitations. These limitations include:

1. It is questionable whether the techniques exist to produce such systems for any but a highly restricted set of domains.
2. Even for those domains in which such systems can be produced the overheads are prohibitive for most purposes.

Given these limitations, I argue that knowledge representation should be utilised in CAL only for those aspects of the instructional process for which it results in substantial gains without prohibitive overheads. I demonstrate that one aspect of CAL for which this holds is for managing flow of control within instructional material.

I provide a detailed description of feature networks. These are a variant of M.A.K. Halliday's system network formalism. Feature networks are a knowledge representation formalism that efficiently encodes exactly the knowledge that is required for knowledge-based flow of control.

It is shown that computer based lessons that utilise feature networks for control flow of control are extremely economic in terms of both authoring time and computer resources while providing highly responsive tuition.

DABIS, a system that embodies the methodology outlined above, has been implemented and is described.

DABIS demonstrates the feasibility of this methodology. There are no fundamental restrictions to the domains to which it can be applied. A lesson created under the DABIS system has been found to have an authoring to student time ration of approximately 12:1. Lessons created by the system also demonstrate the sensitivity to a student's knowledge and abilities within a domain that results from the intelligent use of knowledge-based CAL.

1 INTRODUCTION

The predominant approach that has been adopted in Artificial Intelligence based Computer-Aided Learning (AICAL) has been to generate entire instructional sequences directly from a domain knowledge base. That is to say, most AICAL systems use Artificial Intelligence (Al) techniques for all aspects of instruction.

Almost all of these systems use a pedagogical strategy which I call monitor-remediate tutoring. This strategy entails several stages in the instructional process. First, the student is set a task to perform. The student's performance is monitored and aberrations from optimal performance receive remediation. Depending upon the sophistication of the tutor, such remediation may vary from such actions as demonstrating what the optimal action would have been to analysing the underlying cognitive causes of the student's non-optimal behaviour. I provide a detailed discussion of the monitor-remediate tutorial strategy in Webb (1986a).

The monitor-remediate strategy has several fundamental problems. This approach to CAL has tremendous overheads, both computationally and in terms of development time create. More fundamentally, it relies upon the tutorial system having complete knowledge of optimal performance in the domain which is to be taught. This means that the system must be able to generate and recognise not only one but all optimal solution paths in every task that it sets for the student. Inability to do this means that the system is unable to detect when the student's performance is non-optimal and thus is unable to correctly determine when to interrupt the student's activity.

Any attempt to pursue a monitor-remediate tutorial strategy utilising a system that does not have complete knowledge of the domain is bound to lead to disaster. Quite simply, sooner or later a student will take an optimal action that the system does not recognise as optimal. As a result the system will provide remediation and 'teach' the student that their optimal action was not optimal. Webb (1986a) provides an example where this has happened in practice with the Carnegie-Mellon GREATERP tutor (Anderson, Boyle and Reiser, 1985; Reiser, Anderson and Farrell, 1985).

It may be thought that providing complete knowledge of expertise in a domain is a restriction that AICAL can live with; that it only requires that designers of AICAL systems work a little harder than might otherwise be necessary. However, this is not the case. The implementation of an expert system for a non-trivial domain is an extremely difficult task. Such implementation only requires the codification of one optimal solution path for each problem that the system will handle. Most non-trivial domains contain many significant variations on the solution paths that may be followed to produce optimal results. For many of these domains there is no means available by which to determine whether any particular characterisation of optimal performance is complete. The exceptions to these limitations appear to be domains that operate according to readily formalised principles. The prime example of an AICAL system that successfully exploits these properties of a domain is the SOPHIE system for teaching electronic circuit analysis (Brown, Burton and deKleer, 1982). However, not many domains can be completely characterised in the same manner as SOPHIE's electronic circuit.

It may be thought that it is possible to avoid these problems by adopting an approach to AICAL other than monitor-remediate tutoring. However, no other such approach has emerged.

Perhaps an obvious alternative would be to monitor the student's progress for performance that the system can identify as non-optimal rather than assuming that any action that the system cannot identify as optimal was non-optimal. The obvious means in which to do this may appear to be to incorporate in the system a library of buggy rules in the style of Burton's (1982) DEBUGGY system. Such a system stores not only a knowledge-base containing correct knowledge for the domain, but also a knowledge-base of incorrect or buggy 'knowledge' for the domain. From the student's performance it is possible to deduce the buggy rules that the student has adopted.

However, this implies that it is possible to induce that a rule or other element of knowledge has been used in formulating an action without recourse to the entire body of knowledge that is being utilised. This is simply not possible in many cases. An individual rule will only be valid in the context of the rest of a body of knowledge to which it belongs. Unless the system already knows the rest of the student's knowledge base in many cases it will have no sound basis for deducing that any particular set of rules have contributed in the production of a particular action. Most actions could be produced by any of an infinite number of possible combinations of rules.

Similarly, an element of knowledge can only be considered buggy in the context of a particular body of knowledge. For example, the operation ‘multiply the dividend by 10 divided by the divisor’ will be erroneous in most codifications of expertise in division. However, if it is coupled with a rule that ensures that the result is shifted right by one decimal place, then it can form part of an extremely efficient method of solving some division problems. (If in doubt about this try solving 126583/1.25 first by the standard mode of division and then by the method outlined above!) Indeed, in the example that I provide in Webb (1986a) of GREATERP preventing the student from producing an optimal solution, GREATERP identifies the student’s action (which is actually optimal) as resulting from the application of a buggy rule. In short, a buggy knowledge-base is not the solution to the dilemma.

Even for those domains for which the monitor-remediate tutoring strategy is feasible, it is extremely dubious whether the enormous developmental costs are repaid by courseware that is in any way superior to that which will be developed by alternative means for similar outlays. Programmed learning may be an impoverished paradigm in which to operate but, with sufficient development expenditure, very sophisticated courseware can be produced.

The monitor-remediate strategy has grown out of the generative CAL paradigm. As I argue in Webb (1986b) generative CAL is a particular form of a more general paradigm, courseware abstraction. Courseware abstraction involves the creation of courseware that operates by applying a general treatment of a domain to successive examples from that domain. In generative CAL, these examples are generated by the CAL system and thus the system must be able to analyse the examples from general principles. Courseware abstraction does not require that the system should generate the examples. An alternative is for the course author to explicitly list the examples that the system is to use. In this case, it is also possible to list with each example a description of how it is to be treated. As a result it is not necessary for the system to be able to analyse the examples from general principles. Rather, the system need only embody the knowledge necessary to treat an example given that a basic analysis of the example has already been provided by the lesson author.

As a result, the knowledge-base can be far less sophisticated. This makes its development far less onerous.

It also becomes practical to utilise the AI component of the system only for some aspects of the lesson. The entire instructional sequence need not be directly generated from the knowledge-base.

Much of the power and intuitive appeal of AICAL arises from its ability to respond to the students understanding of a domain and general cognitive needs and to adapt its instruction accordingly. This attribute of AICAL does not directly arise from the use of monitor-remediate tutoring or from the generation of the instructional sequence directly from the knowledge-base.

Why then not utilise AI techniques only for this aspect of the instructional process and use the more traditional and less expensive techniques of programmed learning for all other aspects of instruction? By developing a simple knowledge-base that describes only those aspects of a domain that are required to enable instruction to be adapted to the student’s understanding of a domain and general cognitive needs, most of the problems associated with the monitor - remediate strategy can be avoided.

Consider, if you will, an arbitrary computer-based lesson as a set (possibly infinite) of possible instructional actions. By instructional actions I mean low level input and output operations such as “display the string ‘WHILE is not the name of a recursive function in LISP’ in the top left hand corner of the screen”. The problem at hand is to select the best sequence of instructional actions for a particular student’s knowledge, aptitude and inclination. This involves at any point in time selecting, on the basis of the past history of the student’s actions, the best instructional action from the set of those available. Viewed in this manner, the problem becomes one of flow of control within the set of possible instructional actions.

As has been discussed above, the monitor-remediate strategy is not suitable for generating instructional actions except in domains which can be fully formalised. No other methodology has been proposed for utilising AI to generate instructional activities, so what I wish to propose is that we should return to the tried and true methods of test and branch CAL for the provision of instructional actions.

The use of AI should be reserved for the aspect at which it excels, adapting the instructional sequence to fit the student's needs. In the context that I am proposing, this entails using AI to manage flow of control within a collection of test and branch instructional sequences. That is, an AI based global manager should determine flow of control.

Several requirements are indicated if this is to be achieved. First, if an intelligent system is to select instructional actions on the basis of a student's cognitive needs then it must have a means of determining what cognitive needs a particular low level instructional action will satisfy. Second, the system must have a means of determining what the student's cognitive needs are.

The only satisfactory means of evaluating the student's cognitive needs is by analysing the student's performance as a lesson progresses. A stance in favour of student evaluation from outside a lesson being used to manage flow of control within a lesson would be quite untenable.

Within the architecture that I am proposing, the only sensible locus of evaluation of a student's performance at a particular task is by the test and branch instructional sequence that directly supervises that task. In other words, each test and branch instructional action must return to the AI based global manager an evaluation of the student's performance.

For its part, the global AI manager must understand the domain of knowledge to be taught and how each instructional interaction relates to that domain. The system s knowledge of the domain must be sufficient to be able to evaluate the student's understanding of the domain and to determine how the different aspects of the domain relate to one another. This requires far less complexity in the knowledge-base than if it is to be used to directly generate the instructional sequences.

Feature networks, a variant of Halliday's (1973) system networks, provide exactly the form of knowledge representation that such an AI manager requires. The key elements of feature networks are features and feature choices. A feature represents a property that instances of the domain can exhibit, or alternatively, a predicate that can be applied to instances of the lesson. Every feature belongs to exactly one feature choice. A feature choice describes a dimension of categorisation in a domain. The features in a dimension of categorisation must be both necessarily disjoint and exhaustive. That is, it must necessarily be true that for all instances from the domain to which a dimension of categorisation applies exactly one feature from the future choice applies. Further, a dimension of categorisation represents the epistemological relatedness of the features that it contains. Thus, whereas Red and Blue may be features in a dimension of categorisation for a domain, Red and Square are unlikely to be, even if they are necessarily disjoint and exhaustive for the domain.

Feature networks also contain several mechanisms for specifying how feature choices relate to one another. Webb (1985b) contains a formal description of feature networks and their semantics.

Instances from a domain may be related to the feature network for the domain by a feature set. A feature set contains the features from the domain that the instance exhibits.

Standard test and branch CAL routines can be associated with each aspect of the feature network for a lesson. Such a routine is called a tutorial specification.

A tutorial specification provides an abstract treatment of the aspect of the domain with which it is associated. When a tutorial specification is applied to a particular instance from the domain it generates a concrete treatment of how the aspect of the domain applies to that instance. A tutorial interface at a feature choice has the responsibility of identifying whether the student believes one of the features from that feature choice applies to the instance and if so which one.

Following the courseware abstraction paradigm, the author creates separately the general CAL treatment of the domain and the set of instances from the domain to which it is to be applied. In this case, the general CAL treatment of the domain includes both a feature network that describes the domain and tutorial specifications for each aspect of that network.

The supervisory component of the system manages flow of control at two levels. When examining an instance from a domain it must determine which tutorial specification to invoke and the order in which they are to be invoked. At a higher level, it must determine which instances to examine and the order in which to examine them.

Given that the features for an instance have been specified, the feature network formalism encodes all information necessary to determine which aspects of the general treatment of the domain apply to that instance. Thus, the system is able to determine which aspects of the network are relevant to an examination of the instance and thus which tutorial specifications to invoke in CAL treatment of it. The other manner in which feature networks serve to aid the management of flow of control at the lower level is by specifying epistemological relationships between aspects of the domain. This enables the system to judge which aspects of the domain should be regarded as pre-requisites for which other aspects of the domain. In this way the system is able to ensure that the student is not asked to examine aspects of the domain for which s/he is not likely to have requisite understanding.

At the higher level of flow of control, feature networks enable the system to identify in general which aspects of the domain the student experiences difficulty in examining. This enables the system to produce an abstract description of the types of properties that the student requires more tuition in examining. The system is able to use this abstract description to select concrete instances the examination of which will be of greatest benefit to the student.

By these three mechanisms the system is able to provide adaptive instruction that is directly focussed on the student's individual cognitive needs.

The other major advantage of AICAL is the manner in which it is able to produce cognitive analyses of the student's understanding of the domain. In most AICAL systems these analyses take the form of attributing a subset of the system's knowledge-base to the student. Some systems incorporate incorrect as well as correct 'knowledge' in their knowledge-base thus extending the possible beliefs, skills and other elements of knowledge that can be attributed to the student. The fundamental flaw with such an approach is that the CAL system's knowledge-base must include every aspect of knowledge that it can be relevant to attribute to the student.

By contrast, feature networks characterise properties from a domain and their relationships. The student model is created by recording the instances of the domain and thus the exact combinations of properties for which the student experiences difficulty in analysing. Further, for each time a property is not correctly identified, the property that is attributed to the instance in its place is known to the system. From this record the system is able to produce by statistical analysis a model of how the student's comprehension of the domain differs from its own. This model can include details such as the dimensions of categorisation that the student has not mastered; properties that the student does not understand; instances with which the student is not familiar; and properties that the student cannot identify.

The use of the feature network knowledge representation formalism to manage flow of control is able to produce the adaptive instruction and cognitive models that characterise other AICAL systems. However, they enable this to be done without the tremendous developmental and computational overheads associated with other such systems. The Domain-Analysis Based Instruction System (Webb, 1986c) is an implementation of the methodology spelt out above that has been developed to demonstrate this point.

The largest lesson that has been developed using this system examines the English word class system for Linguistics students. This lesson took six hours to develop. The average terminal time that students have spent on the system is in excess of thirty minutes. This means that there is better than a 12:1 ratio between teacher development time and student terminal time. This compares very favourably with ratios of between 100:1 and 200:1 which are frequently quoted for the development of programmed learning CAL material, let alone the usual costs of AICAL material which can only be guessed at.

2 CONCLUSION

Existing AICAL has that advantage over other approaches that the lessons adapt to the student's cognitive needs and produce cognitive evaluations of the student's performance. However, they are developmentally and computationally expensive and, more fundamentally, require complete expertise in the domain.

As a result, I have argued for eliminating the direct generation of instructional interactions from the knowledge base and instead using it to manage flow of control and student evaluation. This leads to the elimination of most of the developmental and computational overheads of the approach. Further, the approach does not require complete expertise in the domain. These overheads are removed without losing either of the major advantages of the original approach.

References

[1] Anderson, John R., Boyle, Franklin and Reiser, Brian J. (1985). **Intelligent Tutoring Systems**. *Science*, 228:456-462.

[2] Brown, John Seely, Burton, Richard R. and deKleer, Johan. (1982). **Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III**. In Sleeman, D. and Brown, J. S. (eds) *Intelligent Tutoring Systems*. Academic Press, London, 227-307.

[3] Burton, Richard R. (1982). **Diagnosing Bugs in a Simple Procedural Skill**. In Sleeman, D. and Brown, J. S. (eds) *Intelligent Tutoring Systems*. Academic Press, London, 1982, 157-183.

[4] Halliday, M.A.K. (1973). *Explorations in the Functions of Language*. Edward Arnold, London.

[5] Reiser, Brian J, Anderson, John R. and Farrell, Robert G. (1985). **Dynamic student modelling in an intelligent tutor for LISP programming**. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 8-14.

[6] Webb, Geoffrey I. (1986a). **Artificial Intelligence Based Computer Aided Learning Systems**. To be published.

[7] Webb, Geoffrey I. (1986b). **Courseware Abstraction**. To be published.

[8] Webb, Geoffrey I. (1986c). **Feature Networks as a Basis for Computer-Aided Learning**. To be published.

[9] Webb, Geoffrey I. (1986d). **The Domain-Analysis Based Instruction System**. To be published in the *Proceedings of the 1986 CALITE Conference*, Adelaide.