Artificial Intelligence Review, (1990) **4**, 251–277

Intelligent Tutoring Systems: an overview

Hyacinth S. Nwana
Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK

Abstract. This is a non-expert overview of Intelligent Tutoring Systems (ITSs), a way in which Artificial Intelligence (AI) techniques are being applied to education. It introduces ITSs and the motivation for them. It looks at its history: its evolution from Computer-Assisted Instruction (CAI). After looking at the structure of a 'typical' ITS, the paper further examines and discusses some other architectures. Several classic ITSs are reviewed, mainly due to their historical significance or because they best demonstrate some of the principles of intelligent tutoring. A reasonably representative list of ITSs is also provided in order to provide a better appreciation of this vibrant field as well as reveal the scope of existing tutors. The paper concludes, perhaps more appropriately, with some of the author's viewpoints on a couple of controversial issues in the intelligent tutoring domain.

1 Artificial Intelligence and Education

The incorporation of Artificial Intelligence (AI) techniques into education in order to produce educationally useful computer artefacts dates back to the early 1970s. By the early 1980s researchers in the already vibrant field had clearly split into two unequal camps with the emergence of two schools of thought. The first and smaller of the two groups advocated 'exploration environments' — environments which encourage discovery learning (i.e. learning by doing). Perhaps the most famous is the *LOGO* language (Papert, 1980) which introduces students to the world of geometry through the use of robot 'turtles' and 'turtle graphics' techniques, i.e. the student learns by direct programming rather than by indirect instruction. Papert (1980) projects that 'computer presence will enable us to modify the learning environment outside the classroom so that much, if not all, of the knowledge schools presently try to teach with such pain, expense and limited success, will be learned as the child learns to talk, painlessly, successfully and without instruction.' He goes on to conclude that 'schools as we know them today have no place in the future.' Clearly, Papert's dream is quite revolutionary: hence, he and his advocates in the *LOGO* camp are often referred to as revolutionaries.

The second and larger of the camps is the 'intelligent tutoring' group who refer to themselves as reformists as they prefer a gradual improvement (i.e. evolution) in the present quality of education using AI techniques. They advocate a paradigm where the computer acts as a tutor, i.e. students largely learn by being told.

Naturally, the approach Papert champions was bound to be greeted with considerable scorn and criticism as it proposes radically to change the status quo. It appears inconceivable that such a rapid change is feasible even if it were thought desirable. On the other hand, the intelligent tutoring approach enjoys the privilege of being closer to current traditional classroom instruction. As a result, the *LOGO* camp, perhaps unfortunately, is now often bracketed with its much larger intelligent tutoring counterpart. In any case, this paper overviews the latter (i.e. intelligent tutoring systems).

2 Introduction to intelligent tutoring systems

Intelligent tutoring systems (ITSs) are computer programs that are designed to incorporate techniques from the AI community in order to provide tutors which know *what* they teach, *who* they teach and *how* to teach it. AI attempts to produce in a computer behaviour which, if performed by a human, would be described as 'intelligent'; ITSs may similarly be thought of as attempts to produce in a computer behaviour which, if performed by a human, would be described as 'good teaching' (Elsom-Cook, 1987). The design and development of such tutors lie at the intersection of computer science, cognitive psychology and educational research; this intersecting area is normally referred to as cognitive science (see Fig. 1). For historical reasons, much of the research in the domain of educational software involving AI has been conducted in the name of 'ICAI', an acronym for 'Intelligent Computer-Aided Instruction'. This phrase, in turn, evolved out of the name 'Computer-Aided Instruction' (CAI) often referring to the use of computers in education. Nevertheless, to all intents and purposes, ITSs and ICAI are synonymous. However, though some researchers still prefer 'ICAI' (e.g. Self, 1988a, uses it in the title of his recent book), it is now often replaced by the acronym 'ITS' (Sleeman & Brown, 1982b). The latter, which is also the author's personal preference, is certainly gaining support, as confirmed by the international conference on Intelligent Tutoring Systems held in Montreal, Canada, as recently as June 1988. This preference is motivated by the claim that, in many ways, the significance of the shift in research methodology goes beyond the adding of an 'I' to CAI (Wenger, 1987). However, some researchers are understandably hesitant to use the term 'intelligent', instead opting for labels such as 'Knowledge-Based Tutoring System' (KBTS) or 'Adaptive Tutoring System' (ATS) (e.g. Streitz, 1988); Wenger (1987) prefers the label Knowledge Communication Systems. Nevertheless, most researchers appear to be reasonably content with the acronym ITS. This is fine as long as everyone involved with the area understands that the usage of the word 'intelligent' is, strictly speaking, a misnomer. This does not appear to be the case, resulting in some very ambitious goals/claims, particularly in the more theoretical parts of the literature: this also appears to be a valid criticism of the entire AI literature.

The fact that ITS research spans three different disciplines has important implications. It means that there are major differences in research goals, terminology, theoretical frameworks, and emphases amongst ITS researchers. This will become apparent later in this paper. ITS research also requires a mutual understanding of the three disciplines involved, a very stressful demand given the problems of keeping abreast with even a single discipline today.

However, some researchers have stood up to the challenge. As a result, a great deal has been learnt about how to design and implement ITSs. A number of impressive ITSs described in this paper bear testimony to this fact.

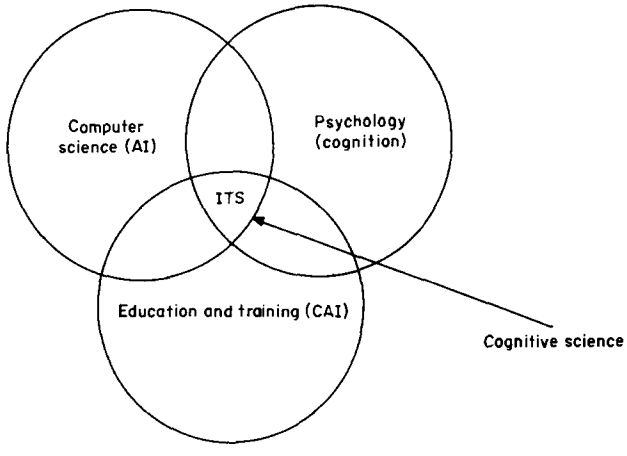

Fig. 1. ITS domains.

3 Motivation

Why do researchers bother to produce such computer-based tutors? There seem to be two main motivating factors.

(1) *Research needs.* On the pure research level, there is a need to understand more about the processes which contribute to an educational interaction (Elsom-Cook, 1987). Since ITS research lies at the intersection of three main disciplines, it provides an excellent test-bed for various theories from cognitive psychologists. AI scientists and educational theorists. For example, a primary reason why the famous Carnegie Mellon psychologist John Anderson came into the area, was to test out his various theories of learning (Anderson, 1987). Hence, the design of an ITS will contribute to the discovery of more accurate theories of cognition (Burns & Capps, 1988).

(2) *Practical needs.* On the more applied level, there are a number of useful results which can be achieved using ITSs which cannot be achieved with human tutors for economic and social reasons (Elsom-Cook, 1987). A primary advantage of ITSs is the possibility for providing one-to-one tutoring. There is a consensus on the view that individual tuition, tailored to the needs of the student, is the most effective form of educational interaction, at least for most domains. Bloom (1984) in his comparison of private tutoring with classroom instruction of cartography and

probability found that 98% of the students with private tutors performed better than the average classroom student, even though all students spent the same amount of time learning the topics. Anderson *et al.* (1985a, b) also recorded a four-to-one advantage for the private tutor, as measured by the amount of time for students to get to the same level of proficiency. Since our educational systems have, of necessity, become geared towards group teaching, many of the advantages of one-to-one tutoring have been lost. ITSs can provide such tuition without necessarily losing the advantages of the group teaching environment (e.g. by providing one ITS per student in a class), thereby getting the best of both worlds. The ITS could provide immediate feedback to the student on the task being performed. This individualized and immediate feedback is crucial because tutoring is most effective when occurring in direct response to the need of the student.

4 Historical review

4.1 Introduction to review

Computer-assisted instruction/learning (CAI/CAL) has evolved considerably since its inception in the 1950s with Skinnerian type 'linear programs'. This has happened despite being set off in the wrong direction by Skinner's insistence that students' responses could be ignored in linear programs (O'Shea & Self, 1983). The central problem with early systems was that they were unable to provide rich feedback or individualization, because they were not designed to know what they were teaching, who they were teaching or how to teach it. In order to solve this problem, CAI/CAL systems have evolved over the past three and a half decades into what are now usually termed 'Intelligent Tutoring Systems' (ITSs). Although we may still be far from truly intelligent tutoring systems, most would agree considerable progress has already been made.

4.2 From CAI to ITSs: major stages

There were some major stages in the metamorphosis of the linear programs of the 1950s into the ITSs of the 1980s (see Fig. 2). The path has spanned a period of almost four decades. It began in the 1950s with simple 'linear programs' which were based on the principle of operant conditioning. The main proponent of such linear programs was the psychologist B. F. Skinner (1954, 1958). Material which had been selected and arranged to take the student step by step towards the desired behaviour was presented in a series of 'frames'. Most frames had very simple questions (e.g. involving only the filling in of a missing space or two), and the student was told immediately whether the answer was right or wrong. The system proceeded to present the next frame regardless of the correctness of the student's response. To be fair, Skinner held that students should not be allowed to make mistakes because this gives negative reinforcement. If the designer succeeded in this aim, all the responses would be correct and so could legitimately be ignored. Unfortunately, experience showed that such an ideal situation was usually not attainable. The major limitations of linear programs then became glaringly appa-

rent: they did not provide individualization, which meant that all students, irrespective of their abilities, background, or previous knowledge of the domain, received exactly the same material in exactly the same sequence; neither did they provide feedback, as the students' responses were ignored. This style of CAI has been dubbed *ad-hoc* frame-orientated (AFO) CAI by Carbonell (1970) to stress its dependence on author-specified units of information. Carbonell concluded that 'in most CAI systems of the AFO type, the computer does little more than what a programmed text book can do, and one may wonder why the machine is used at all when teaching sequences are extremely simple, perhaps trivial, one should consider doing away with the computer, and using other devices or techniques more related to the task' (Carbonell, 1970, pp. 194, 201). Overcoming these limitations prompted the chain of events which has culminated in today's ITSs.

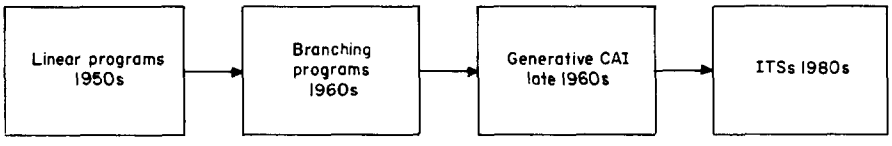

Fig. 2. CAI to ITS metamorphosis.

Crowder (1959) overcame some of the limitations of Skinnerian systems by ceasing to ignore students' responses. He proposed using them to control the material shown to the student. The 'branching programs' that resulted still had a fixed number of frames, but were able to comment on a student's response and then use it to choose the next frame, possibly repeating an earlier one. Pattern-matching techniques allowed alternate answers to be treated as acceptable or partially acceptable rather than as totally correct or incorrect as demanded by Skinnerian systems. However, the teaching material became too large to be manageable through straightforward programming and so a special breed of programming languages, called 'author languages', were developed for creating CAI material.

In the late 1960s and early 1970s, 'generative systems' came into being (also called 'adaptive systems'). These emerged from the recognition of the fact that the teaching material could itself be generated by the computer. A generative system has the ability both to generate and solve meaningful problems. In some domains like arithmetic, researchers realized they could do away with all the pre-stored teaching material, problems, solutions and associated diagnostics, and actually generate them. The potential advantages, if exploited, were enormous. They included drastically reduced memory usage and the generation and provision of as many problems (to some desired level of difficulty) as the student needed. Most notably, Uhr and his team implemented a series of systems which generated problems in arithmetic that were 'tailor-made' to a student's performance (Uhr, 1969). Suppes (1967) and Woods & Hartley (1971) produced systems with similar abilities. Wexler (1970) describes a system which combines generative CAI with frame-orientated CAI in which the course-author must specify certain question formats. The system generates parameters for these formats and searches the database to determine the correct answer. Nevertheless, a major shortcoming was the

restriction to drill-type exercises in domains as well-structured as mathematics. Only parametric summaries of behaviour were used to guide problem generation, rather than an explicit representation of the student's knowledge (Sleeman & Brown, 1982b). Sleeman (1983) notes that in the initial version of the Leeds Adaptive Arithmetic System (which later evolved to the Leeds Modelling System, LMS, Sleeman & Smith (1981) and from there to the PIXIE system (Sleeman, 1987)), the model of the student consisted merely of an integer to indicate the level of the student's competence.

Generative CAI was the main precursor of ITSs. Although individualization and feedback had been improved, there was a rather shallow knowledge representation. Yazdani (1986) notes that 'none of these systems has human-like knowledge of the domain it is teaching, nor can it answer the serious questions from the students as to "why" and "how" the task is performed.' Such sentiments have also been echoed by other researchers (e.g. O'Shea & Self, 1983). Hence, many problems remained unsolved. Sleeman & Brown (1982b) and Hawkes *et al.* (1986) point out that these systems were found to be lacking for a number of reasons.

(1) They attempted to produce total courses rather than concentrating on building systems for more limited topics.
(2) They had severe natural language barriers which restricted users' interaction with them.
(3) They had no 'knowledge' or 'understanding' of the subject they tutored or of the students themselves; this is sometimes referred to in the literature as the 'Eliza' syndrome. Consequently, they tended to assume too much or too little student knowledge, and they could not conceptualize so as to diagnose a student's misconception within his/her own framework.
(4) They were extremely *ad hoc*. Building tutoring systems was not recognized to be a non-trivial task — a task requiring detailed psychological theories of learning and mislearning. Anyone with a knowledge of computing attempted to build a tutor. Consequently, there was little or no co-operation among educators, psychologists and computer scientists in the development phase of the tutors.
(5) They tended to be static rather than dynamic. There was little experimentation with systems in order to improve them. Human tutors learn about their students and about the subjects they teach every day, and so should machine tutors.

In response to the problems CAI faced, Self, in his interesting and classic paper, argued that a computer tutorial program should have a representation of *what* is being taught, *who* is being taught and *how* to teach him/her (Self, 1974). Carbonell (1970) argued that a solution to this problem could not be achieved without the use of AI techniques. Jaime Carbonell's important contribution to cognitive science is best summarized in the title of his first-rate 1970 publication *AI in CAI*. He wanted to put AI into CAI systems. He dreamed of a system which had a database of knowledge about a subject matter and general information about language and principles of tutorial instruction. The system could then pursue a natural language dialogue with a student, sometimes following the student's initiative, sometimes taking its own initiative, but always generating its statements and responses in a natural way from its general knowledge. Such a system sharpedly contrasted with

existing CAI systems at the time in which a relatively fixed sequence of questions and possible responses had to be pre-determined for each topic. He constructed an early version of his dream, a classic system he called *SCHOLAR* (Carbonell, 1970, 1971), but he died before it was fully realized. Carbonell's introduction of AI into CAI marked the beginning of the era of ITSs which therefore emerged to provide answers to the limitations of generative CAI. (ITSs are alternatively referred to as Intelligent Computer-Assisted Instruction/Learning (ICAI/ICAL) systems as mentioned in Section 2.) It is claimed that ITSs combine AI, psychological models of the student and the expert, and educational theory. The psychological models allow for simulations of student performance and can be experimented upon until they closely represent the behaviour exhibited by the student.

Summarizing, providing a truly 'intelligent' system was recognized to be a non-trivial task which needed experts from several other disciplines; a need which could be provided by the AI research community, which contains computer scientists, psychologists and educationalists. Most of the present-day work in ITSs is

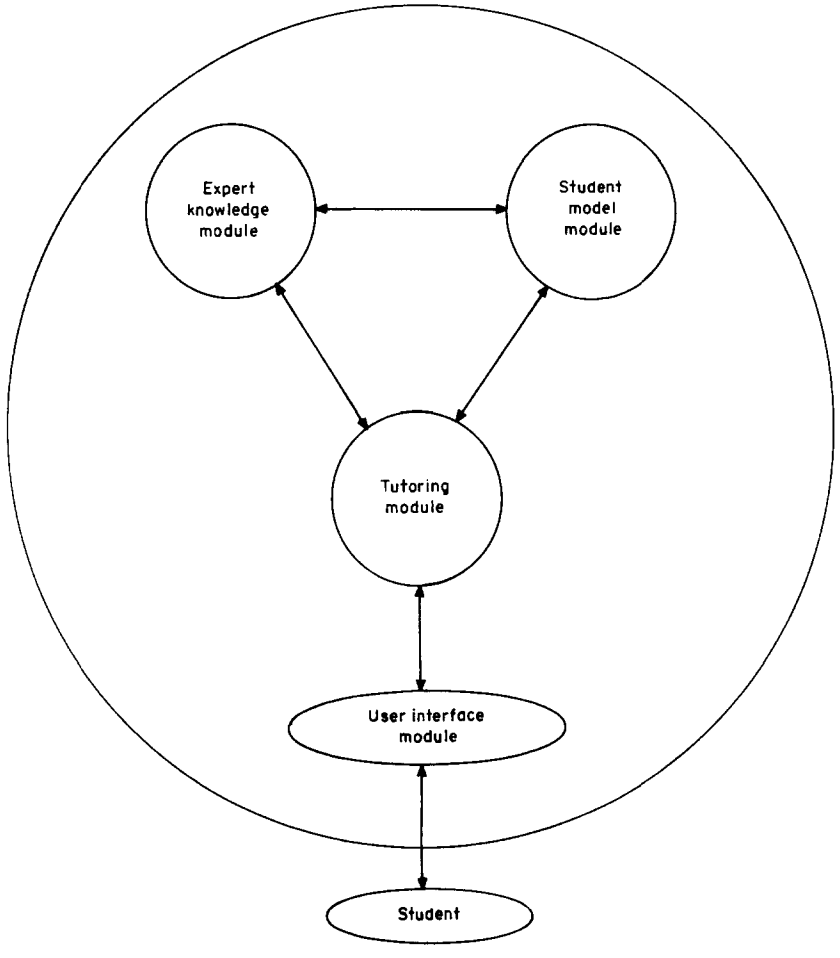

Fig. 3. General ITS architecture.

being carried out by AI researchers or enthusiasts, and AIs influence has been so great that Yazdani (1983) concludes that 'intelligent tutoring systems are AI's answer to CAL packages.' However, it is worth making the point that most so-called ITSs still do not escape the bulk of the criticisms discussed above, so that there is considerable scope for further work. Some of the major differences between ITSs and CAI programs are (or should be):

(1) ITSs provide a clear articulation of knowledge for a limited domain;
(2) ITSs have a model of student performance which is dynamically maintained and is used to drive instruction;
(3) the ITS designer defines the knowledge and the inference rules, but *not* the teaching sequence, which is derived by the program;
(4) ITSs provide detailed diagnostics of errors rather than simply drill and practice;
(5) students can pose questions to an ITS (this is the main characteristic of 'mixed-initiative tutors').

5 Structure of ITSs

5.1 *General Architecture*

Existing ITSs vary tremendously in architecture. In fact, it is almost a rarity to find two ITSs based on the same architecture. This results from the experimental nature of the work in the area: there is yet no clear-cut general architecture for such systems (Yazdani, 1986, 1987). Previously, there was considerable consensus in the literature that ITSs consist of at least three basic components (Barr & Feigenbaum, 1982; Bonnet, 1985).

(1) The expert knowledge module.
(2) The student model module.
(3) The tutoring module.

However, more recent research (Wenger, 1987; Burns & Capps, 1988; Mandl & Lesgold, 1988) has identified and added a fourth component to the list.

(4) The user interface module.

Figure 3 illustrates the general form of an ITS architecture, which will serve as a basis for discussion. It does not represent any particular known system.

The expert knowledge module comprises the facts and rules of the particular domain to be conveyed to the student, i.e. the knowledge of the experts. In the transition from CAI to ITSs, such knowledge has been the first aspect of the teacher's expertise to be explicitly represented in systems. It has already been mentioned that in traditional CAI, the expertise to be communicated is contained in pre-stored presentation material called 'frames', which are designed by the expert teacher and simply displayed to the student under given conditions. Such implicit representation of knowledge has since been recognized to be inadequate: in fact a major lesson learned from all the research on expert systems is that any expert module must have an abundance of specific and detailed knowledge, derived from

people who have years of experience in a particular domain. Consequently, in ITSs, much effort is expended in discovering and codifying the domain knowledge, i.e. distilling years of experience into a knowledge representation.

Knowledge elicitation and codification can be a very time-consuming task, especially for a complex domain with an enormous amount of knowledge and interrelationships of that knowledge. Thus, investigating how to encode knowledge and how to represent it in an ITS remains the central issue of creating an expert knowledge module. In effect, this process aims to make the knowledge stored in this module more explicit. So, in current ITSs, expert knowledge is represented in various ways, including semantic networks, frames and production systems. It must not only include surface knowledge (e.g. the descriptions of various concepts that the student has to acquire), but also the representational ability that has been recognized to be a critical part of expertise. Expert knowledge must include the ability to construct implicit representational understanding from explicitly represented information (Mandl & Lesgold, 1988).

Expert knowledge modules can be classified along a spectrum ranging from completely opaque or 'blackbox' representations, whereby only final results are available (e.g. in tutors like *SOPHIE I*, reviewed in the next section), to fully transparent or 'glassbox' ones, where each reasoning step can be inspected and interpreted (e.g. *SOPHIE III*, also reviewed later).

The expert knowledge module or domain expert, as it is alternatively termed, fulfils a double function. First, it serves as the source of knowledge to be presented to the student, which includes generating questions, explanations and responses. Secondly, it provides a standard for evaluating the student's performance. For this latter function, it must be able to generate solutions to problems in the same context as the student, so that respective answers can be compared. The module must also be able to detect common systematic mistakes, and if possible identify any gap in the student's knowledge that may be the cause of this. If the ITS is to monitor students in solving problems, the expert module must also be able to generate sensible, and possibly multiple, solution paths so that intermediate steps can be compared.

Also, in its function as a standard, the expert knowledge module can be used to assess the student's overall progress. To achieve this required the establishment of some criteria to compare knowledge. This type of comparison is possible only if the knowledge has been explicitly represented. Hence, ITSs differ considerably from traditional CAI programs in that the knowledge in the latter is implicitly represented within its code.

It is also worth recognizing that the expert knowledge module by necessity embodies a specific view of the domain — that of the designer. Thus, tutoring can be compromised if the student does not understand the system's instruction or because the system cannot interpret the student's behaviour in terms of its own view of the knowledge (Wenger, 1987). Of course, human teachers also have their own views, but they do also have the incredible ability to adapt them accordingly in order to perceive that of the student. This issue touches on the central problem of knowledge representation in AI; a problem which is often said to be the main

bottle-neck to AI's success. The solution to this bottle-neck is particularly relevant to ITSs because of its deep pedagogical implications.

The student model module refers to the dynamic representation of the emerging knowledge and skill of the student. No intelligent tutoring can take place without an understanding of the student. Thus, along with the idea of explicitly representing the knowledge to be communicated came the idea of doing likewise with the student, in the form of a student model. Most researchers agree that an ITS should have a student model (Hartley & Sleeman, 1973; Self, 1974, 1987a, 1987b, 1988b; Rich, 1979; Sleeman, 1985; Tobias, 1985; Zissos & Witten, 1985; Ross *et al.*, 1987; Gilmore & Self, 1988; Wachsmuth, 1988). Ideally, this model should include all those aspects of the student's behaviour and knowledge that have possible repercussions on his/her performance and learning. However, the task of constructing such a complete model is not only non-trivial but, probably, impossible, especially considering that the communication channel, which is usually the keyboard, is so restrictive. Human tutors would normally combine data from a variety of other sources, like voice effects or facial gestures. They may also be able to detect other phenomenological factors such as boredom or motivation which are also crucial in learning.

In a recent review, Self (1988b) identified twenty different uses that had been found for student models in existing ITSs. From analysing this list, he notes that the functions of student models could be generally classified into six types.

(1) Corrective: to help eradicate bugs in the student's knowledge.
(2) Elaborative: to help correct 'incomplete' student knowledge.
(3) Strategic: to help initiate significant changes in the tutorial strategy other than the tactical decisions of 1 and 2 above.
(4) Diagnostic: to help diagnose bugs in the student's knowledge.
(5) Predictive: to help determine the student's likely response to tutorial actions.
(6) Evaluative: to help assess the student or the ITS.

In the author's view, Self's preceding list is still far from comprehensive; it could be narrowed down much further, and the student model could be seen to fulfil a double function. On the one hand, it acts as a source of information about the student. On the other hand, it serves as a representation of the student's knowledge. Wenger (1987) also supports this view. The next few paragraphs will attempt to justify this aforementioned viewpoint.

In its function as a source of information, it infers unobservable aspects of the student's behaviour from the model. Such an inference could produce an interpretation of his/her actions and also lead to a reconstruction of the knowledge that gave rise to these actions. Such knowledge is vital for the pedagogic component of the ITS as it could be used in either of the six ways noted by Self.

The student model is also likely to be formed out of the system's representation of the target knowledge in the expert knowledge module. Accordingly, the student model can include a clear evaluation of the mastery of each unit of knowledge in the expert module (the function of the student model here is evaluative). This

allows the student's state of knowledge to be compared with the expert knowledge module, and instruction would then be biased towards portions of the model shown to be weak (thus, the student model's function here is elaborative). This form of student modelling is referred to as 'overlay' modelling (Goldstein, 1982), because the student's state of knowledge is viewed as a subset of the expert's. Again, this shows how the student model acts as a source of information.

However, incorrect or suboptimal behaviour does not always result from incomplete knowledge. It could also be due to incorrect versions of the target knowledge. Therefore a more formative student model should also provide explicit representations of the student's incorrect versions of the target knowledge for remediation purposes (clearly, the function of the student model here is diagnostic and corrective). In such a capacity, it serves as a representation of the student's knowledge. This approach to modelling is called the 'buggy' approach (Brown & Burton, 1978a).

Student models are also expected to be executable or runnable. This allows for exact prediction about a particular student in a particular context (thus, the function of the student model here is predictive). The tutoring module would also be making use of such executable representations for pedagogic purposes (i.e. the knowledge represented in the student model is being used in a strategic way). The student model is also serving here as a representation of the student's knowledge.

In conclusion, student models could be seen to perform two 'super' functions: acting as a source of information about the student, and serving as a representation of the student. In achieving these functions, they act in roles including corrective, elaborative, strategic, diagnostic, predictive and evaluative.

The tutoring module is the part of the ITS that designs and regulates instructional interactions with the student. In other architectures, this module is referred to as the teaching strategy or the pedagogic module. It is closely linked to the student model, using knowledge about the student and its own tutorial goal structure to decide which pedagogic activities will be presented: hints to overcome impasses in performance, advice, support, explanations, different practice tasks, tests to confirm hypotheses in the student's model, etc. (Self, 1988b). The tutorial component is thus the source and the orchestrator of all pedagogic interventions. The decisions involved are subtle. The order and manner in which topics are treated can produce very different learning experiences. In tutorial, it is sometimes more effective to let the student flounder for a while before interrupting; sometimes, the student will get stuck or lost if left completely to himself/herself (however, no good human tutor will destroy a student's personal motivation or sense of discovery). Consequently, the tutoring in existing ITSs can be classified along a spectrum ranging from systems that *monitor* the student's every activity very closely, adapting their actions to the student's responses but never relinquishing control, to *guided-discovery learning* systems where the student has almost full control of the activity, and the only way the system can direct the course of action is by modifying the environment. In the middle are *mixed-initiative* systems where the control is shared by the student and the system as they exchange questions and answers. The existence of this spectrum clearly highlights the fact that tutoring is an art that

requires great versatility which is still extremely difficult to articulate and represent in an ITS. Nevertheless, some progress is being made, albeit limited.

ITSs also aim at explicitly representing the knowledge found in the tutoring module. This creates the potential to adapt and improve strategies over time (as in the case of self-improving tutors), and for the same strategies to be used for other domains. Once more this is contrasted with traditional CAI systems where such pedagogical knowledge is deeply buried in the various pieces of code that control the tutorial interaction.

The user interface module is the communicating component of the ITS which controls interaction between the student and the system, as depicted in Fig. 3. In both directions, it translates between the system's internal representation and an interface language that is understandable to the student. Because the user interface can make or break the ITS, no matter how 'intelligent' the internal system is, it has become customary to identify it as a distinct component of its own. In fact, it would be a mistake to consider it a secondary component of the ITS for two main reasons. First, when the ITS presents a topic, the interface can enhance or diminish the presentation. Since the interface is the final form in which the ITS presents itself, qualities such as ease of use and attractiveness could be crucial to the student's acceptance of the system. Secondly, progress in media technology is increasingly providing more and more sophisticated tools whose communicative power heavily influences ITS design.

Current ITSs provide user interfaces which, for the input, range from the use of fixed menus with multiple-choice answers to a fairly free treatment of a pseudo-natural language. For the output, they range from the mere display of pre-stored texts typical of CAI, to the use of fairly complicated generic frames. Within these two ends of the spectrum there is also a varying flexibility (Wenger, 1987).

Some ITSs are making their interaction with the student more 'user friendly' by substituting pictures and pointing for text and typing. In some, the treatment of text is only supplemented by pictures and graphics; much more cognitive research into the use of such interfaces is still required. However, the most one expects to see, in the near future, is ITSs communicating with the student primarily via graphics, as research in natural language understanding and computer speech recognition/generation are still at such early and primitive stages. In fact, research into user interfaces is just commencing, as they have only recently been acknowledged as distinct parts of ITSs.

5.2 *Other architectures*

Other architectures have been suggested, some largely similar, but others radically different to that depicted in Fig. 3. For example, there are four components to Anderson's Advanced Computer Tutoring (ACT) ITS architecture (Anderson *et al.*, 1985a,b) (see Fig. 4).

(1) The domain expert (ideal student model): this module contains all the correct rules used for solving problems in the domain.
(2) The bug catalogue: this is an extensive library of common misconceptions and errors for the domain.

(3) The teaching knowledge (tutoring module).
(4) The user interface.

This architecture is not only a proposal; at least two tutors are based on it including the Geometry tutor and the Lisp tutor (Anderson *et al.*, 1985a).

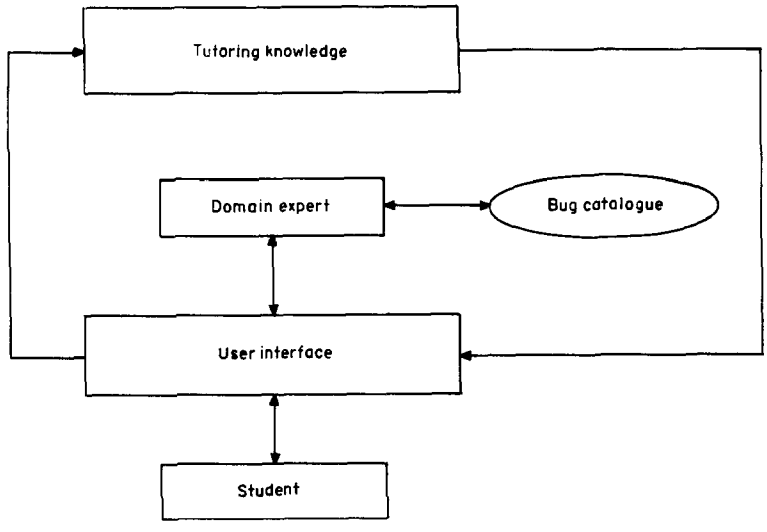

Fig. 4. Andersons ITS architecture.

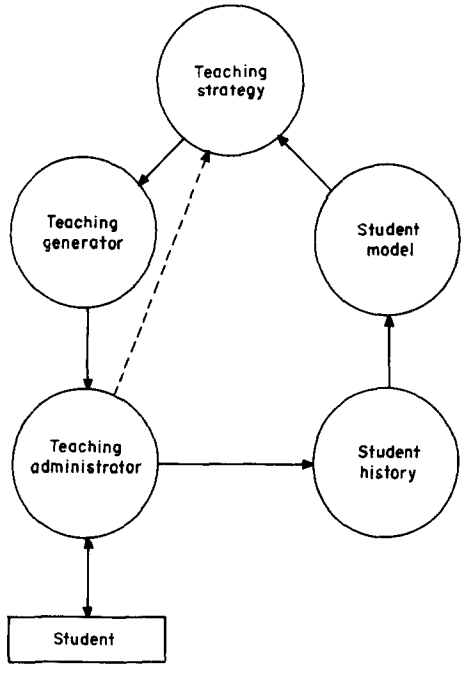

Fig. 5. O'Shea *et al.*'s architecture.

Hartley & Sleeman, whose 1973 architecture is probably the closest proposal to the general architecture depicted in Fig. 3, suggest that an ITS ought to have four distinct knowledge bases:

(1) knowledge of the domain (expert knowledge),
(2) knowledge of the person being taught (student model),
(3) knowledge of the teaching strategies (tutoring knowledge),
(4) knowledge of how to apply the tutoring knowledge to the needs of an individual.

Hartley & Sleeman's proposals differ from Anderson's inasmuch as they do not give the misconceptions in the domain (the bug catalogue) primary importance but instead introduce the student model as a primary component (Yazdani, 1987). Furthermore, this proposal subsumes the user interface in a more tutoring-orientated module which includes meta-rules that guide the tutoring rules. These differences also reflect the different tutoring philosophies involved in both architectures; Anderson plays down the importance of the student model and, in its place, substitutes two knowledge bases of ideal and buggy representations of knowledge of the domain (see Fig. 4). Immediately a behaviour is exhibited which indicates an error or bug, the student is nudged to follow the correct path by being presented with the ideal solution.

O'Shea *et al.* (1984) present a five-ring model as shown in Fig. 5. This bears some similarity to the Hartley & Sleeman architecture. However, it also clearly demonstrates how differences in emphasis on student modelling and teaching lead to an architecture which is starkly different from Anderson's (see Fig. 4). Its components include:

(1) student history,
(2) student model,
(3) teaching strategy,
(4) teaching generation,
(5) teaching administration.

In this proposal, the explicit representation of knowledge in the domain (expert knowledge), and the common misconceptions in the domain (the bug catalogue), are undermined in favour of teaching skills: hence the introduction of the teaching generation and teaching administration components.

Figure 6 presents another architecture which introduces the self-improving (learning) concept. The architecture thus forms the basis of self-improving systems which attempt to improve their tutoring capabilities over time. Such systems are still rare and typically consist of two components: an adaptive teaching program (which may have any of the architectures discussed earlier or some other), and a self-improving component that makes experimental changes using data collected during teaching sessions. O'Shea's (1982) Quadratic tutor was one of the earliest to exploit this idea. This ambitious architecture, which attempts to automate a process that is even difficult for humans, is an important contribution; most researchers agree but pay lip service to the fact that ITSs should improve or learn over time as

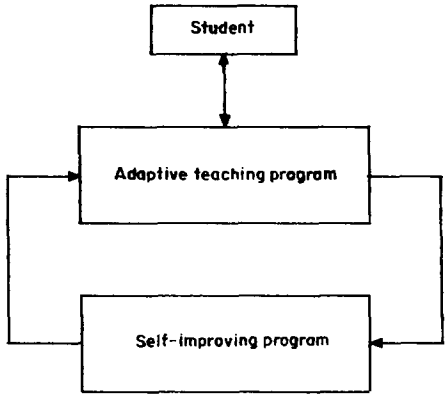

Fig. 6. Self-improving ITS architecture.

good human tutors do. Kimball's (1982) Integration tutor is generally acknowledged to be the first self-improving tutor to have been developed.

In addition to the above architectures, more recent systems concentrate on other features such as planning (e.g. see Peachey & McCalla, 1986). The SCENT-3 architecture evolved by McCalla's team (McCalla & Greer, 1988) is much more complex and interested readers are referred to this paper. Suffice to say that it is based on a blackboard philosophy; the blackboard mediates communication among the numerous components of the architecture.

5.3 *Discussion*

It is thus evident that differing tutoring philosophies place emphases on different aspects of the instructional process (e.g. student modelling), and this in turn leads to different architectures, as in the case of O'Shea *et al.* and Anderson. It is inconceivable that there would be a consensus amongst researchers on a tutoring philosophy; rather, more variations of present philosophies are emerging. Consequently, this has resulted in the numerous ITS architectures in the literature. Naturally, not all these architectures can support a range of tutoring strategies within a given ITS.

Most of these architectures still remain only proposals. Also, almost all the arguments to their support or as to which of them is the 'best' have, disappointingly, been theoretical rather than practical. Until ITSs built around these proposals are evaluated and demonstrated to be educationally worthy, none can claim any superiority over another. Regrettably, a significant fraction of these proposals have not even had systems developed which are based on them. Admittedly, a few have (e.g. Anderson's), but an insufficient number of ITSs based on these architectures have been developed, evaluated and proven 'useful' to warrant such arguments. In fact, due to the experimental nature of the area, even the so-called general architecture of Fig. 3 should be viewed with some degree of scepticism and should definitely not be seen as a basis for all ITSs. When more developed ITSs become evaluated, as is expected to be the case, it is possible that some additional, or

perhaps alternate, set of building blocks might emerge (as is the case in the blackboard architecture of McCalla & Greer, 1988).

6 Review of some classic ITSs

In this section, some important ITSs are reviewed; an overview paper of this sort would be incomplete without it. It is by no means meant to be exhaustive: rather, it should augment the previous sections of this paper. The systems reviewed are either chosen for their historical significance or because they are good examples of systems which display some of the intelligent tutoring principles reported in this paper.

6.1 *SCHOLAR*

As the system which successfully launched the new paradigm of intelligent tutoring, *SCHOLAR* surely deserves a place in any review of ITSs. Needless to say it was the first ITS to be constructed. It was a revolutionary system when considered in its historical context; most of the then existing ITSs were of the *ad-hoc* frame-orientated (AFO) type. *SCHOLAR* was created by Jaime Carbonell; this automatically earned him a place in history as founder of ITSs. He used *SCHOLAR* to launch a new paradigm which he called 'information-structure-oriented' (ISO) CAI as opposed to the predominant AFO-type CAI of the time (Carbonell, 1970, 1971). These two approaches correspond in many ways to what are now respectively called traditional CAI and ITS. Because *SCHOLAR*, like most other ITS projects, evolved, the version described in this review is Carbonell's original version: besides, it is the important one because of its historical significance to ITS research.

SCHOLAR was a pioneering effort in the development of computer tutors capable of handling unanticipated student questions and of generating instructional material in varying levels of detail, depending on the context of the dialogue. It was a mixed-initiative ITS; both the system and the student could initiate conversation by asking questions. Both the program's output and the student's inputs were English sentences. It seems appropriate to review it further by considering its four components, as defined in Section 5.1.

The knowledge in the expert knowledge module is that of the geography of South America, which was represented in a semantic network whose nodes instantiated geographical objects and concepts. Statements like 'Tell me more about Brazil' just invoked a retrieval of facts stored in the semantic network. However, the real power of this representation schema comes by recognizing that it is possible to answer questions for which answers are not stored. This automatically relieves the system of the memory problems encountered in anticipating and storing all solutions by traditional CAI systems. For example, it is not necessary to store in the semantic network that 'Lima is in South America' provided that the program which interprets the network can make the relevant inference. In other words, the program must know about the attributes concerned, e.g. 'location' and 'capital', and in particular, that if x is capital of y and y is located in z then x is in z: this is a rule of inference.

A semantic network representation was chosen because Carbonell thought it to be close to the teacher's conceptualization of knowledge. By implication, the network also represented the ideal student's conceptualization (ideal student model): hence, overlay modelling becomes feasible. So *SCHOLAR* could associate flags with each node of the network to indicate whether the student was thought to know the information represented by that node. More ambitiously, Carbonell proposed to model student errors by introducing small 'perturbations' to the network; this proposal was not followed up in *SCHOLAR*. Its student modelling was then extremely rudimentary.

SCHOLAR's tutorial strategies were also fairly primitive, consisting mainly of local topic selections. The teacher using it was expected to provide an agenda. Whenever a topic was too general, *SCHOLAR* generated a subtopic on an essentially random basis. For example, the teacher might specify the topic of 'South America', and *SCHOLAR* would select a subtopic, e.g. 'Peru', and then perhaps a sub-subtopic, e.g. 'topography of Peru'. This random element led to somewhat disconnected discussions lacking the systematic development of ideas characteristic of a good tutorial, though it was necessary since *SCHOLAR*'s semantic network had little information about desirable orders of presentations of topics. Nevertheless, suitable relevant tags in the network (from the agenda), could provide *SCHOLAR* with some reasonable guidance in selecting topics.

SCHOLAR possessed language processing capabilities that were also rather limited. Text was generated by sentence and question templates that was filled up with information from the network. The parsing of student's questions followed the same principle in reverse, while the parsing of student's answers was done by matching key words from a list dynamically generated from the network for each question. Consequently, *SCHOLAR* did not understand wrong answers and so could not glean diagnostic information from them.

SCHOLAR has not been widely used except in *NLS-SCHOLAR*, an intelligent on-line consultant for a text editor (Grignetti *et al.*, 1975). This was partly due to some fundamental limitations such as difficulty of representing procedural knowledge using semantic nets. However, despite all its shortcomings, *SCHOLAR* introduced many methodological principles that have become central to ITS design, e.g. separation of tutorial strategies from domain knowledge, more explicit representation of knowledge, student modelling, etc. Indeed, *SCHOLAR*'s significance as a milestone for the entire field cannot be over-emphasized.

6.2 SOPHIE

SOPHIE (a SOPHisticated Instructional Environment) is an ITS which reflects a major attempt to extend Carbonell's notion of mixed-initiative CAI (introduced in *SCHOLAR*) for the purpose of encouraging a wider range of student initiatives (Brown *et al.*, 1975). It was developed by John Seely Brown, Richard Burton, and their coleagues at Bolt Beranek and Newman, Inc. This project went through successive phases spanning more than 5 years; the three stages of development of *SOPHIE* (*I–III*) incorporate the most intensive attempt at building at complete ITS

so far. *SOPHIE* is a milestone for the field, and hence it well earns its place in this review.

The pedagogic philosophy is different in *SOPHIE*: it is not so much to imitate a dialogue with a human teacher (as *SCHOLAR* sought to do) as to provide a reactive learning environment in which the student can try his ideas, have them assessed, and receive advice. Its philosophy is thus 'learning by doing' as opposed to 'learning by being told' as in the case of *SCHOLAR*. Brown *et al.* (1982) suggest that computer technology can be used to make experimentation both 'easier' and 'safer' by simulating environments that capitalize on the motivational value of exploratory problem-solving activities. *SOPHIE*'s simulated area of expertise is electronic troubleshooting. Since the components of a simulation can be made faulty, troubleshooting means performing a series of measurements to propose and test hypotheses concerning the location and nature of the fault. This not only gives the student the opportunity to apply a theoretical knowledge of electronic laws, but also to acquire general troubleshooting strategies. In essence, it enables the student to have a one-to-one relationship with an 'expert' who helps create, experiment with, and debug his/her own ideas (Brown *et al.*, 1975). In keeping with the objectives of this paper, *SOPHIE* will further be reviewed by considering the four components as defined in Section 5.1.

The expert knowledge module of *SOPHIE* comprises a 'strong' model (simulation) of electronic troubleshooting for the IP-28 regulated power supply and a 'canned' articulate expert troubleshooter which can not only solve problems, but is also capable of explaining its tactics and high-level strategies for attacking the problem. For example, the expert can explain 'why' a measurement was made and 'what' logically follows from the measurement obtained.

SOPHIE's tutoring module possesses numerous heuristic strategies for answering and critiquing a student or generating alternative theories to his/her current hypotheses. *SOPHIE I*'s tutorial capabilities were impressive as mentioned in the previous paragraph; however, with the evolution from *SOPHIE I* to *SOPHIE III*, the strategies were enhanced to become more 'human-like'. This was because its implementors observed that *SOPHIE I* and *II*'s approaches to problem solving were foreign to humans (Brown *et al.*, 1982). Consequently, *SOPHIE* evolved from a simulation-based inference system to a more powerful and human-like reasoning system (using qualitative reasoning techniques). By implication, *SOPHIE*'s sophisticated tutoring module requires a similarly sophisticated student model.

By all accounts, the *SOPHIE* interface demonstrates a very impressive natural language capability; it uses the powerful notion of semantic grammars proposed by Richard Burton. It is robust (handling 'nearly all sentences generated by users who have had a few minutes exposure to the system'), efficient ('understands a typical statement in a fraction of a second'), and of some generality ('since the notion of semantic grammars has been successfully applied to other areas besides electronics'); the interface undeniably demonstrates that techniques for processing natural language are sufficiently developed to be usable in ITSs (O'Shea & Self, 1983).

Summarizing, *SOPHIE*'s performance as a 'complete' ITS is presently unsurpassed. *SOPHIE* was actually commissioned by the American Defence Department and had limited use for on-site job training over the ARPA network (ARPA Internet is a

network of several networks; the most important ARPANET links US research centres and universities (with 150 nodes)) for 2 years; it is no longer maintained (no reason has been given in the literature for this). However, considering that most other prototype ITSs never ever get used after their development, *SOPHIE* was quite a success story. Probably, *SOPHIE*'s most important contribution to ITS, though, was to establish it as a respectable subarea in the eye of the AI community (Wenger, 1987).

6.3 *GUIDON*

GUIDON, an ITS for teaching diagnostic problem solving, was developed by William Clancey and his colleagues at Stanford University. The *GUIDON* project is also unique as it represents the first attempt to adapt a pre-existing expert system into an intelligent tutor. Probably because it was strongly influenced by *SCHOLAR* and *SOPHIE*, it turned out to be one of the most concerned efforts so far at designing an ITS. Like the other two, it went through many stages spanning more than 5 years during which it yielded many important findings. All of this coupled with the fact that it is built around the most well-known expert system, *MYCIN*, earns it a place as one of the key ITS projects ever undertaken.

GUIDON's goal is to tutor the knowledge from the famous expert system, *MYCIN* (Shortliffe, 1976), a medical expert system that suggests treatment for bacterial infections. It attempts to transfer expertise to the students exclusively through case dialogues where a sick patient (the 'case') is described to the student in general terms. The student is then asked to play the role of a physician and ask for information he/she thinks might be relevant to the case. *GUIDON* compares the student's questions to those which *MYCIN* would have asked and critiques him/her on this basis; this demonstrates a different tutoring strategy to that of *SCHOLAR* or *SOPHIE*. It is easy to infer from the previous sentence that student modelling is largely of the overlay-type. *GUIDON* also separates its tutorial strategies (comprising 200 rules), which was largely influenced by *SOPHIE*'s, from its domain knowledge. Nevertheless, its natural language capabilities are far less sophisticated than *SOPHIE*'s, but certainly improve on *SCHOLAR*'s.

The component which has evolved considerably has been the expert module. The original version (*GUIDON 1*) was implemented by 'reversing' *MYCIN*'s 450 rules (Clancey, 1984). This implementation was ineffectual largely because medical diagnosis is not made 'cookbook' style — i.e. medical practitioners do not diagnose diseases by using perfect recall on hundreds of medical facts and rules (Clancey, 1982, 1983, 1987). He realized that *MYCIN*'s rules represent 'compiled' knowledge devoid of the low-level detail and relation necessary for learning and tutoring. *GUIDON 1*'s failure was largely due to the fact that it would have had to 'decompile' and augment these rules with data and diagnostic hypotheses that the medical practitioner uses implicitly. Thus, *MYCIN*'s rules were reconfigured to separate the strategic knowledge from the domain facts and rules, resulting in *NEOMYCIN* (Clancey & Letsinger, 1981). This in turn became the new basis around which *GUIDON 2* was built, and with some improved teaching strategies, *GUIDON 2* has had greater success as a prototype tutoring system than *GUIDON 1* (Clancey, 1987).

The *GUIDON* project provided a fascinating inquiry into the epistemological questions related to intelligent tutoring as well as producing many important findings about designing ITSs. For example, it clearly demonstrated that an expert system is not a sound basis for tutoring (Elsom-Cook, 1987). More importantly however, *GUIDON* produced spectacular demonstrations of the field's ability to bring to light fundamental AI research issues (Wenger, 1987).

6.4 *WEST*

The *WEST* coach is a program developed, again, by Richard Burton and John Seely Brown to help students play a game on the *PLATO* system (*PLATO* was one of the largest ever CAI projects ever undertaken). It is indeed an intelligent tutor but in view of the implementation of this program in an informal learning environment, the term 'coach', originated by Goldstein (1982), appeared more congenial than 'tutor' (Burton & Brown, 1977, 1982). *WEST* was a spin-off from the *SOPHIE* project; hence, it is still in keeping with the concept of a reactive learning environment central to *SOPHIE*, but requires much simpler skills. *WEST* is also the first ever computer coach and it demonstrates how a different emphasis on different components of the ITS (in this case the tutoring module) can produce a radically different ITS, so radical that the term 'coach' is preferred to 'tutor'.

WEST simulates a board game requiring players to travel in a series of moves. The number of spaces for each move are determined by digits on three 'spinners' supplied from a random-number generator. Players can combine these three digits by using any legitimate mathematical operation including exponentiation, or by using negative numbers, parentheses, etc. The game also has such features as short cuts to the goal; opportunities to 'bump' opponents, forcing them to return to the beginning, and spaces safe from bumping. Although two students can play against each other, they typically play against the machine. At each move, the student's skill in writing algebraic equations is compared to the expert's solution for the same skill. If the two solutions differ, the coach (tutoring component) can intervene and provide the student with helpful hints as to how to improve his/her game or make better moves.

WEST's expert knowledge module comprises the simulated board game and an articulate expert which can monitor and evaluate the student's moves. The student modelling technique is largely a simpler version of overlay modelling called differential modelling. This is because, apart from outright arithmetical errors, the student's moves are never wrong; they are just poor. What is important is the difference in comparison between the expert's move and the student's: hence the word 'differential'. *WEST*'s interface is simple as its inputs are mainly arithmetical expressions involving integers (e.g. $1 + 2 \times 2$) or just plain integers.

However, the component that makes *WEST* radically different from the previous three systems discussed is its tutoring component. Its main strategy is to encourage skill acquisition and general problem-solving abilities by engaging the student in some game-like activity. In effect, the immediate aim is to have fun; skill acquisition and learning is an indirect consequence.

WEST has actually been used in elementary school classrooms. In a controlled experiment, a coached group exhibited 'a considerably greater variety of patterns' in the expressions they formed and they even 'enjoyed playing the game considerably more than the uncoached group' (Burton & Brown, 1982). These results are quite encouraging: they demonstrate that the coach succeeded in fostering learning without any apparent adverse effect on the fun of the game. Unfortunately, coaching currently appears only to be applicable to trivial domains; it is hard to see how it could be used to teach, say, electronic troubleshooting or some non-obvious fraction addition problem.

However, *WEST*'s influence on ITSs has been significant, and it is still a reference for researchers today. Regrettably, it seems it is no longer used. With so many worthless computer games available on the market, it is really unfortunate that a program like *WEST*, which has actually been demonstrated to be functional, should still remain a laboratory prototype (Wenger, 1987).

7 The range of prototype ITSs

This section, as in Ross (1987), lists examples of ITSs that have been developed. The list of ITSs and environments shown in Table 1 below is reasonably representative and provides an appreciation of the vibrant nature of this new and interesting field and also reveals the scope of already constructed systems. As also echoed by Ross (1987) most of them do a lot less than the domain indication suggests and are also mostly experimental as hardly any have been tested on more than a very few people. The list also reveals that the domains chosen by ITS researchers are mainly few. Nevertheless, with the experience being currently accrued by researchers, more complex domains are expected to be tackled in future research.

8 Some viewpoints

It is appropriate to conclude this paper with some viewpoints on some of the contentious issues in the intelligent tutoring domain; it also has its own on-going debates. Two pertinent ones are:

(1) is intelligent tutoring just old wine in a new bottle, or is it a new vintage (Ok-choon *et al.*, 1987)?
(2) is intelligent tutoring really possible (Ridgway, 1988)?

Before providing responses to these questions, it seems fair to remark that they arise in the first place because of public opinion about the AI enterprise in general. Bobrow *et al.* (1986) correctly point out that public opinion about AI is schizophrenic, ranging from 'it will never work' to 'it might cost me my job'. This range of opinion reflects the collective confusion about AI. It appears that the AI fraternity only has itself to blame. Misleading publicity, mostly in order to attract grants, and misuse of flamboyant expressions like artificial intelligence, expert systems, intelligent tutoring systems, etc., have helped contribute to unrealistic expectations of the state-of-the-art. Consequently, questions such as those above are bound to be asked.

Table 1. A reasonably comprehensive list of ITSs and environments

ITS	Domain	Reference
ACE/PSM	NMR spectra interpretation	Sleeman (1975)
ATDSE	Basic subtraction	Attisha & Yazdani (1983)
ARITHMEKIT	Basic subtraction	Brown (1983)
ALGEBRALAND	Algebraic proofs	Brown (1985)
BIP-I/BIP-II	Basic programming	Barr *et al.* (1976)
BLOCKS Tutor	Troubleshooting in a *BLOCKS* world	Brown & Burton (1978b)
BRIDGE	Programming	Bonar (1985)
BUGGY	Basic subtraction	Brown & Burton (1978a)
DEBUGGY	Basic subtraction	Burton (1982)
EDSMB	Basic multiplication	Attisha & Yazdani (1984)
EUROHELP	*UNIX* mail	Breuker (1987)
EXCHECK	Basic logic	Blaine (1982)
FGA	Basic French grammar	Barchan *et al.* (1986)
FITS	Basic fractions addition	Nwana (1990)
FLOW Tutor	*FLOW* computer language	Genter (1977)
GEOMETRY Tutor	Geometry proofs	Anderson *et al.* (1985a)
GERMAN Tutor	Basic German	Weischedel *et al.* (1978)
GUIDON I/II	Basic medical diagnosis	Clancey (1987)
INTEGRATION Tutor	Basic integral calculus	Kimball (1982)
LISP Tutor	Lisp programming	Anderson & Reiser (1985)
LMS	Basic algebra	Sleeman & Smith (1981)
MACSYMA Advisor	Use of *MACSYMA*	Genesereth (1982)
MALT	Basic machine language programming	Koffman & Blount, 1975
MENO-Tutor	Basic *Pascal* programming	Woolf & McDonald (1984)
METEOROLOGY ITS	Basic meteorology	Brown *et al.* (1973)
NEOMYCIN	Medical diagnosis	Clancey & Letsinger (1981)
PIXIE	Basic algebra	Sleeman (1987)
PROUST	*Pascal* programming	Soloway & Johnson (1984)
QUADRATIC Tutor	Quadratic equations	O'Shea (1982)
QUEST	Basic electrics	White & Frederiksen (1985)
SCENT-3 Advisor	*Lisp* programming	McCalla *et al.* (1988)
SCHOLAR	South American geographical facts	Carbonell (1970)
SIERRA	Learning basic arithmetic procedures	Vanlehn (1987)
SOPHIE I/II/III	Basic electronic troubleshooting	Brown *et al.* (1982)
SPADE	Basic *LOGO* programming	Goldstein & Miller (1976)
SPIRIT	Probability theory	Barzilay (1985)
STEAMER	Marine steam propulsion plant	Hollan *et al.* (1984)
TALUS	Basic *Lisp* programming	Murray (1987)
THEVENIN	Basic electrical circuits	Joobbani & Talukdar (1985)
TUTOR	British highway code	Davies *et al.* (1985)
WEST	Basic arithmetic skills	Brown & Burton (1978b)
WHY	Basic meteorology	Collins & Stevens (1982)
WUSOR	Maze game skills	Goldstein (1982)

In the first question, the old wine was basically the AFO-type CAI which has been observed to have taken a very naive view of the instructional process. It has since been realized that providing a truly 'intelligent' tutor is a non-trivial task requiring experts from several disciplines. ITSs combine at least some of the following: AI, psychological models of the student and expert, and educational theory. There is thus a substantial change in the quality of wine and, therefore, a new vintage.

The second question investigates whether intelligent tutoring is really possible. Many answers to such questions in the past have turned out to be far too optimistic. For example, Suppes' (1966) prediction that 'in a few more years millions of school children will have access to what Phillip of Macedon's son had as royal prerogative: the personal services of a tutor as well-informed as Aristotle.' Clearly, besides the obvious financial constraints forbidding this and without much deliberation on what Suppes meant by 'few', this prediction is still far from being achieved a quarter of a century on. Hence, the response to this second question requires more caution. The answer seems to depend on how the questioner views the use of the word 'intelligent' in AI terms. If he/she rightly views it as strictly speaking a misnomer, at least for now, then intelligent tutoring becomes very feasible, at least in various limited domains. On the other hand, if the questioner is more literal, as many AI sceptics in the literature seem to be (again probably due to the misleading publicity), then of course intelligent tutoring would appear impossible, and so would AI in general. Nevertheless, there is a suspicion that no matter how 'intelligently' an ITS may eventually perform (e.g. even if one were demonstrably to perform beter than most human tutors), these sceptics would still not be satisfied, mainly because ITSs are computer-based. It therefore appears that researchers will eventually have to turn to the famous Turing test of intelligence for an answer when ITSs come of age. This is because Turing's test circumvents the problem of lack of consensus on the definition of the word 'intelligent': it regards intelligence as undefinable but 'intelligent behaviour' as recognizable.

However, one prediction looks secure: that despite these controversies, ITS research will grow. This is because, apart from their practical needs, the area appears to provide an excellent test-bed for theories from AI scientists, educational theorists and cognitive psychologists (for instance, it has been noted that the Carnegie Mellon psychologist John Anderson came into the area to test his psychological theories). Furthermore, ITS researchers themselves would be very interested in incorporating any promising research results from various AI/Cognitive Science subdomains, e.g. qualitative reasoning, planning, natural language understanding, human–computer interaction, etc., into their ITSs, and so they should. In conclusion, it appears certain that the best of ITS research is yet to come.

9 Conclusions

This paper has introduced the AI subdomain of intelligent tutoring systems and the motivation of ITSs. It looked at its history, motivation as well as reviewed some notable ITSs. The paper also reveals how different emphases on different components of the ITS (because of different philosophies adopted by various researchers) result in contrasting systems; it provides examples of different architectures to support this. This also reflects the different disciplines which ITS spans. The paper also lists a reasonably comprehensive set of prototype ITSs which have been developed to date. The paper draws to a conclusion with the author's viewpoints on two debatable issues in the ITS domain.

For further reading, see Sleeman & Brown (1982a), O'Shea & Self (1983), Wenger (1987) or Nwana (1989).

Acknowledgments

The author sincerely acknowledges the invaluable comments of his supervisor, Dr Peter Coxhead and of his examiner, Dr Peter Ross.

References

Anderson, J. R. (1987) Production systems, learning and tutoring. In *Production System Models of Learning and Development* (eds D. Klahr, P. Langley & R. Neches). MIT Press, London, pp. 437–458.

Anderson, J. R., Boyle, D. G. & Reiser, B. J. (1985a) Intelligent tutoring systems. *Science*, **228**, 456–462.

Anderson, J. R., Boyle, D. F. & Yost, G. (1985b) The geometry tutor. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 1–7.

Anderson, J. R. & Reiser, B. J. (1985) The lisp tutor. *Byte*, **10**(4).

Attisha, M. G. & Yazdani, M. (1983) A micro-computer based tutor for teaching arithmetic skills. *Instructional Science*, **12**, 333–342.

Attisha, M. G. & Yazdani, M. (1984) An expert system for diagnosing children's multiplication errors. *Instructional Science*, **13**, 79–92.

Barchan, J., Woodmansee, B. J. & Yazdani, M. (1986) A prlog-based tool for French grammar analysis. *Instructional Science*, **14**.

Barr, A., Beard, M. & Atkinson, R. C. (1976) The computer as a tutorial laboratory: the Stanford BIP Project. *International Journal of Man-Machine Studies*, **8**, 567–596.

Barr, A. & Feigenbaum, E. A. (1982) *The Handbook of Artificial Intelligence*, **Vol. 2.** Kaufmann, Los Altos.

Barzilay, A. (1985) SPIRIT: a flexible tutoring style in an intelligent tutoring system. In *Artificial Intelligence Applications: The Engineering of Knowledge-Based Systems* (ed. R. C. Weisbin). IEE Computer Society, North Holland.

Blaine, L. H. (1982) EXCHECK. *Handbook of Artificial Intelligence*, Vol. 2 (eds A. Barr & E. A. Feigenbaum). Addison-Wesley, Reading, MA.

Bloom, B. S. (1984) The 2 Sigma Problem: the search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher* **13**, 4–16.

Bobrow, D. G., Mittal, S. & Steffik, M. (1986) Expert systems: perils and promise. *Communications of the ACM*, **29**, 880–893.

Bonar, J. (1985) *Understanding the Bugs of Novice Programmers*. PhD Thesis, Dept of Computer and Information Science, University of Pittsburgh, Pittsburgh, PA.

Bonnet, A. (1985) *Artificial Intelligence: Promise and Performance*. Prentice Hall, London.

Breuker, J. (1987) Coaching in help systems. In *Artificial Intelligence and Human Learning: Intelligent Computer-aided Instruction* (ed. J. A. Seft). Chapman & Hall, London.

Brown, J. S. (1985) Process versus product: a perspective on tools for communal and informal electronic learning. *Journal of Educational Computing Research* **1**, 179–201.

Brown, J. S. & Burton, R. E. (1978a) Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, **2**, 155–192.

Brown, J. S. & Burton, R. R. (1978b) A paradigmatic example of an artificially intelligent instructional system. *International Journal of Man-Machine Studies*. **10**, 323–339.

Brown, J. S., Burton, R. R. & Bell, A. G. (1975) SOPHIE: a step towards a reactive learning environment. *International Journal of Man-Machine Studies*, **7**, 675–696.

Brown, J. S., Burton, R. R. & de Kleer, J. (1982) Pedagogical, natural language, and knowledge engineering techniques in SOPHIE I, II and III. In *Intelligent Tutoring Systems* (eds D. H. Sleeman & J. S. Brown). Academic Press, London, pp. 227–282.

Brown, J. S., Burton, R. R. & Zydel, F. (1973) A model-driven question-answering system for mixed-initiative CAI. *IEEE Transactions on Systems, Man, and Cybernetics*, **3**, 248–257.

Burns, H. L. & Capps, C. G. (1988) Foundations of intelligent tutoring systems: an introduction. In *Foundations of Intelligent Tutoring Systems* (eds M. C. Polson & J. J. Richardson). Lawrence Erlbaum, London, pp. 1–19.

Burton, R. (1982) Diagnosing bugs in a simple procedural skill. In *Intelligent Tutoring Systems* (eds D. H. Sleeman & J. S. Brown). Academic Press, London, pp. 157–183.

Burton, R. & Brown, J. S. (1977) A tutoring and student modelling paradigm for gaming environments. *SIGCSE Bulletin*, **8**, 236–246.

Burton, R. & Brown, J. S. (1982) An investigation of computer coaching for informal learning activities. In *Intelligent Tutoring Systems* (eds D. H. Sleeman & J. S. Brown). Academic Press, London, pp. 79–98.

Carbonell, J. R. (1970) AI in CAI: an artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, **II**, 190–202.

Carbonell, J. R. (1971) Artificial intelligence and large interactive man computer systems. *Proceedings of the Joint National Conference on Major Systems*, Anahein, CA, pp. 167–173.

Clancey, W. J. (1982) Tutoring rules for guiding a case method dialogue. In *Intelligent Tutoring Systems* (eds D. H. Sleeman & J. S. Brown). Academic Press, London, pp. 201–225.

Clancey, W. J. (1983) GUIDON. *Journal of Computer-Based Instruction*, **10**, 8–15.

Clancey, W. J. (1984) Methodology for building an intelligent tutoring system. In *Methods and Tactics in Cognitive Science* (eds W. Kintsch, J. R. Miller & P. G. Polson). Lawrence Erlbaum, London.

Clancey, W. J. (1987) *Knowledge-Based Tutoring*. MIT Press, London.

Clancey, W. J. & Letsinger, R. (1981) NEOMYCIN: reconfiguring a rule-based expert system for application to teaching. *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouver, Canada, pp. 829–835.

Collins, A. & Stevens, A. L. (1982) Goals and strategies for inquiry teachers. In *Advances in Instructional Psychology*, Vol 2 (ed R. Glaser). Lawrence Erlbaum, Hillsdale, NJ.

Crowder, N. A. (1959) Automatic tutoring by means of intrinsic programming. In *Automatic Teaching: The State of the Art*, Wiley, New York, pp. 109–116.

Davies, N. G., Dickens, S. L. & Ford, L. (1985) Tutor — a prototype ICAI system. In *Research and Development in Expert Systems* (ed. M. A. Bramer). Cambridge University Press, Cambridge.

Elsom-Cook, M. (1987) Intelligent Computer-aided instruction research at the Open University. *Technical Report No: 63*. Computer-Assisted Learning Research Group, The Open University, Milton Keynes.

Genesereth, M. R. (1982) The role of plans in intelligent teaching systems. In *Intelligent Tutoring Systems* (eds D. H. Sleeman & J. S. Brown). Academic Press, London, pp. 137–155.

Genter, D. R. (1977) The FLOW tutor: a schema-based tutorial system. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*. Cambridge, MA 787–790.

Gilmore, D. & Self, J. A. (1988) The application of machine learning to intelligent tutoring systems. In *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction* (ed. J. A. Self). Chapman & Hall, London, pp. 179–196.

Goldstein, I. P. (1982) The genetic graph: a representation for the evolution of procedural knowledge. In *Intelligent Tutoring Systems* (eds D. H. Sleeman & J. S. Brown). Academic Press, London, pp. 51–77.

Goldstein, I. P. & Miller, M. L. (1976) AI-based personal learning environments: directions for long term research. *AI lab Memo 384*. Massachussetts Institute of Technology, Cambridge, MA.

Grignetti, M., Hausman, C. L. & Gould, L. (1975) An intelligent on-line assistant and tutor: NLS-SCHOLAR. *Proceedings of the National Computer Conference*, 775–781.

Hartley, J. R. & Sleeman, D. H. (1973) Towards more intelligent teaching systems. *International Journal of Man-Machine Studies*, **5**, 215–236.

Hasemann, K. (1981) On difficulties with fractions. *Educational Studies in Mathematics*, **12**, 71–287.

Hawkes, W. L., Sharon, J. D., Kandel, A. & Taps Project Staff (1986) Fuzzy expert systems for an intelligent computer-based tutor. *Technical Report No: 86-5*. Learning Systems Institute, Centre for Educational Technology, Florida State University.

Hollan, J. D., Hutchins, E. L. & Weitzman, L. (1984) STEAMER: an interactive inspectable simulation-based training system. *AI Magazine*, **5**, 15–27.

Joobbani, R. & Talukdar, S. N. (1985) An expert system for understanding expressions for electric circuits analysis. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, pp. 23–25.

Kimball, R. A. (1982) A self-improving tutor for symbolic integration. In *Intelligent Tutoring Systems* (eds D. H. Sleeman & J. S. Brown). Academic Press, London, pp. 283–307.

Koffman, E. B. & Blount, S. E. (1975) Artificial intelligence and automatic programming in CAI. *Artificial Intelligence*, **6**, 215–234.

Mandl, H. & Lesgold, A. (eds) (1988) *Learning Issues for Intelligent Tutoring Systems*. Springer-Verlag, London.

McCalla, G. I., Greer, J. E. & SCENT Team (1988) Intelligent advising in problem solving domains: the SCENT-3 architecture. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, Montreal, Canada, pp. 124–131.

Murray, W. R. (1987) Automatic program debugging for intelligent tutoring systems. *Computational Intelligence*, **3**(1).

Nwana, H. S. (1989) *An iterative-style approach to constructing intelligent tutoring systems in mathematics*. PhD Thesis, Aston University, Birmingham.

Nwana, H. S. (1990) The anatomy of FITS: a mathematic tutor. *Intelligent Tutoring Media*, **1**(2).

Ok-choon, P., Ray, S. P. & Seidel, R. J. (1987) Intelligent CAI: old winein new bottles or a new vintage? In *Artificial Intelligence and Instruction: Instruction and Methods*, Addison-Wesley, Reading, MA, pp. 11–43.

O'Shea, T. (1982) A self-improving quadratic tutor. In *Intelligent Tutoring Systems* (eds D. H. Sleeman & J. S. Brown). Academic Press, London, pp. 283–307.

O'Shea, T., Bornat, R., Du Boulay, B., Eisenstadt, M. & Page, I. (1984) Tools for creating intelligent computer tutors. In *Artificial and Human Intelligence* (eds ?. Elithorn & R. Beneiji). Elsevier, North Holland, pp. 181–199.

O'Shea, R. & Self, J. (1983) *Learning and Teaching with Computers*. Harvester Press, Sussex.

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas*. Basic Books, New York.

Peachey, D. R. & McCalla, G. I. (1986) Using planning techniques in intelligent tutoring systems. *International Journal of Man-Machine Studies*, **24**, 77–98.

Rich, E. (1979) User modelling via stereotypes. *Cognitive Science*, **3**, 329–354.

Ridgway, J. (1988) Of course ICAI is impossible . . . worse though, it might be seditious. In *Artificial Intelligence and Human Learning: Intelligent computer-aided instruction* (ed. J. A. Self). Chapman & Hall, London, pp. 28–48.

Ross, P. (1987) Intelligent tutoring systems. *Journal of Computer Assisted Learning*, **3**, 194–203.

Ross, P., Jones, J. & Millington, P. (1987) User modelling in intelligent teaching and tutoring. In *Trends in Computer Assisted Instruction* (eds R. Lewis & E. D. Tagg). Blackwell, London, pp. 32–44.

Self, J. A. (1974) Student models in computer-aided instruction. *International Journal of Man-Machine Studies*, **6**, 261–276.

Self, J. A. (1987a) The application of machine learning to student modelling. In *Artificial Intelligence and Education* **1**: *Learning Environments & Tutoring Systems* (eds R. Lawler & M. Yazdani). Ablex, Norwood, pp. 267–280.

Self, J. A. (1987b) Realism in student modelling. *Alvey-IKBS Research Workshop Tutoring Systems*. University of Exeter.

Self, J. A. (ed.) (1988a) *Artificial Intelligence and Human Learning: Intelligent computer-aided instruction*. Chapman & Hall, London.

Self, J. A. (1988b) Student models: what use are they? In *Artificial Intelligence Tools in Education* (eds P. Ercoli & R. Lewis). North Holland, Amsterdam, pp. 73–86.

Shortliffe, E. H. (1976) *Computer Based Medical Consultations: MYCIN*. Elsevier, New York.

Skinner, B. F. (1954) The science of learning and the art of teaching. *Harvard Educational Review*, **24**, 86–97.

Skinner, B. F. (1958) Teaching Machines. *Science*, **128**, 969–977.

Sleeman, D. H. (1975) A problem-solving monitor for a deductive reasoning task. *International Journal of Man-Machine Studies*, **7**, 183–211.

Sleeman, D. H. (1983) Intelligent tutoring systems: a review. *Proceedings of EdCompCon '83 meeting*. IEEE Computer Society, pp. 95–101.

Sleeman, D. H. (1985) UMFE: a user modelling front-end subsystem. *International Journal of Man-Machine Studies*, **23**, 71–88.

Sleeman, D. H. (1987) PIXIE: a shell for developing intelligent tutoring systems. In *Artificial Intelligence and Education* (eds R. Lawler & M. Yazdani). Ablex, Norwood, pp. 239–265.

Sleeman, D. H. & Brown, J. S. (eds) (1982a) *Intelligent Tutoring Systems*. Academic Press, London.

Sleeman, D. H. & Brown, J. S. (1982b) Introduction: intelligent tutoring systems. In *Intelligent Tutoring Systems* (eds D. H. Sleeman & J. S. Brown). Academic Press, London, pp. 1–11.

Sleeman, D. H. & Smith, M. J. (1981) Modelling students' problem solving. *Artificial Intelligence*, **16**, 171–188.

Soloway, E. & Johnson, W. (1984) Remembrance of blunders past: a retrospective on the development of PROUST. *Proceedings of the Sixth Cognitive Science Society Conference*. Boulder, CO, p. 57.

Streitz, N. A. (1988) Mental models and metaphors: implications for the design of adaptive user-system interfaces. In *Learning Issues for Intelligent Tutoring Systems* (eds H. Mandl & A. Lesgold). Springer-Verlag, London, pp. 164–186.

Suppes, P. (1966) The uses of computers in education. *Scientific American*, **25**, 206–221.

Suppes, P. (1967) Some theoretical models for mathematics learning. *Journal of Research and Development in Education* **1**, 5–22.

Tobias, S. (1985) Computer assisted instruction. In *Adapting Instruction to Individual Differences* (eds M. C. Wang & H. J. Waldberg). McCutchan, Berkeley, CA, pp. 139–159.

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

Vanlehn, K. (1987) Learning one subprocedure per lesson. *Artificial Intelligence* **31**, 1–40.

Wachsmuth, I. (1988) Modelling the knowledge base of mathematical learners: situation-specific and situation-nonspecific knowledge. In *Learning Issues for Intelligent Tutoring Systems* (eds H. Mandl & A. Lesgold). Springer-Verlag, London, pp. 63–79.

Weischedel, R. M., Voge, W. M. & James, M. (1978) An artificial intelligence approach to language instruction. *Artificial Intelligence* **10**, 225–240.

Wenger, E. (1987) *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann, Los Altos, CA.

Wexler, J. D. (1970) Information networks in generative computer-assisted instruction. *IEEE Transactions on Man-Machine Systems* **11**, 181–190.

White, B. Y. & Frederiksen, J. R. (1985) QUEST: qualitative understanding of electrical system troubleshooting. *ACM SIGART Newsletter*, **93**, 34–37.

Woods, P. & Hartley, J. R. (1971) Some learning models for arithmetic tasks and their use in computer-based learning. *British Journal of Educational Psychology*, **41**, 35–48.

Woolf, B. P. & McDonald, D. D. (1984) Context-dependent transitions in tutoring discourse. *Proceedings of the National Conference on Artificial Intelligence*. Austin, Texas, pp. 355–361.

Yazdani, M. (1983) Introduction: artificial intelligence and education. In *New Horizons in Educational Computing* (ed. M. Yazdani). Wiley, New York.

Yazdani, M. (1986) Intelligent tutoring systems survey. *Artificial Intelligence Review*, **1**, 43–52.

Yazdani, M. (1987) Intelligence tutoring systems: an overview. In *Artificial Intelligence and Education* (eds R. Lawler & M. Yazdani). Ablex, Norwood, pp. 182–201.

Zissos, A. Y. & Witten, I. H. (1985) User modelling for a computer coach: a case study. *International Journal of Man-Machine Studies*, **23**, 729–750.