Comp. & Maths. with Appls. Vol. 11, No. 5, pp. 517–526, 1985
Printed in Great Britain.

0097–4943/85 $3.00 + .00

APPLICATIONS OF ARTIFICIAL INTELLIGENCE WITHIN EDUCATION

MARLENE JONES (COLBOURN)
Logic Programming and Artificial Intelligence Group, Department of Computer Science,
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

(*Received September* 1984)

Abstract—Computers have been employed within the field of education for many years, often with disappointing results. However, recent and current research within the field of artificial intelligence (AI) is having a positive impact on educational applications. For example, there now exist ICAI (intelligent computer-assisted instruction) systems to teach or tutor many different subjects; several such systems are discussed herein. In addition to CAI (computer-assisted instruction) systems, we discuss the development of learning environments that are designed to facilitate student-initiated learning. A third major application is the use of expert systems to assist with educational diagnosis and assessment. During the course of our discussion of these three major application areas, we indicate where AI has already played a major role in the development of such systems and where further research is required in order to overcome current limitations.

1. INTRODUCTION

Computers, particularly microcomputers, are now in extensive use within the educational system. In addition to their use in teaching programming skills, they are employed in instructional and diagnostic roles within a variety of subject areas. A computer can also be used as an educational resource (analogous to a library or laboratory), as well as a means for maintaining data bases of student information. Of all the possible uses of computers within education, the application which has received the most attention is the use of the computer within the instructional role.

Despite the proliferation of CAI programs on the market, there are few that truly warrant use within the classroom. The majority of CAI systems suffer from several limitations:

(i) an inability to conduct conversations with the student in the student's natural language;

(ii) an inability to understand the subject being taught, thus being unable to accept unanticipated responses;

(iii) an inability to decide what should be taught next;

(iv) an inability to anticipate, diagnose, and understand the student's mistakes and misconceptions;

(v) an inability to improve or modify current teaching strategies or learn new ones.

Many of these shortcomings correspond to issues that are addressed within major research areas of AI, such as natural language understanding, knowledge representation, planning, expert systems, and learning. Not surprisingly, recent advances in AI are having an impact on the quality of CAI programs currently being developed. We illustrate this point with a brief discussion of particular systems later.

Although the instructional role is the major focus of AI applications in education, it is by no means the only one. In order to provide appropriate instruction, one must first determine what should be taught. This is of particular importance when an individual is experiencing difficulty learning within the regular school environment. Providing teachers with assistance in the diagnosis and assessment of learning difficulties is a major application of computers in education.

This paper is not intended to be a comprehensive survey of all educational software or projects that have employed AI tools or techniques. Rather, we indicate where AI has already had a major impact on the type of system being produced and where further research is needed to overcome current limitations. By way of illustration, we focus our attention on three major application areas: CAI, student-initiated learning, and diagnosis and assessment.

2. INTELLIGENT CAI SYSTEMS

Early CAI programs were little more than drill-and-practice units. In a traditional CAI system the instructor provides the course material to be presented to the student, the criteria for evaluation, and the possible routes through the material to be chosen according to the evaluations. Although various authoring languages, such as PILOT and NATAL, exist to assist the teacher in the development of such CAI systems, they do not overcome the major limitations of traditional CAI systems. Fortunately, many recent CAI systems have gone beyond this in an attempt to provide what Brown[5] calls a *reactive learning environment*, in which the student is actively engaged with the instructional system, and the student's interests and misunderstandings drive the tutorial dialogue. Such systems are often referred to as ICAI (intelligent CAI), knowledge-based CAI, or AICAI systems, since tools and techniques from AI have been employed in order to make such systems both more flexible and sensitive.

Such a CAI system must be capable of analyzing a wide range of student responses; this generally requires the inclusion of a domain expert. In other words, an ICAI system is an expert in the particular topic that it is teaching or tutoring. In addition to this expert component, an ICAI system generally contains a student model that represents both the student's knowledge and misconceptions, as well as a component containing information regarding appropriate teaching strategies. In most ICAI systems the course material is represented independently of both the student model and the instructional procedures or strategies. Another important component of any ICAI system is its interface with the student. Ideally, it should be a conversational system capable of a natural language dialogue.

2.1 *Components of ICAI systems*

Most ICAI systems include a domain expert (in other words, an expert system). A great deal of recent research and attention in AI has focussed upon expert systems; for example, see [3,26,33] and the references therein. An expert system embodies knowledge of a particular application area combined with inference mechanisms that enable the program to employ this knowledge in problem-solving situations[26]. Expert systems have now been developed in a variety of areas, including chemistry, medical diagnosis, geology, computer-system configuration, and mathematics. Within an ICAI system the expert or problem-solving module must contain the domain-specific knowledge; this may be facts about a particular subject and/or information regarding problem-solving skills or procedures relevant to this particular domain. The expert module uses its information about the subject area or domain to generate appropriate questions and to evaluate the student's responses. In some cases, an existing expert system (such as NEOMYCIN[15]) has been later incorporated into the expert module of an instructional system (in this case, GUIDON2[30]).

The second component, the student model, is a method for representing the student's understanding of the course material. Within the model one must represent what the student has mastered as well as difficulties, misconceptions, and unmastered skills. This information must be gleaned from the student's answers, implicit problem-solving behaviour, or perhaps historical information regarding the student's past experiences. Associated with the student model, there must be sophisticated algorithms for determining which skills the student has not mastered; this can be a nontrivial task, especially when the student's difficulty arises on a task requiring multiple skills. Another problem is bridging the gap between the various steps of a student's solution; this may involve conjecturing missing steps or discovering the underlying motives or plans that led to the student's observed sequence of steps.

Many different approaches to student modeling have been proposed and implemented. For example, Goldstein and Carr[24] suggest an *overlay model* in which one hypothesizes which skills of the embedded expert the student possesses. Hence, the structure of the student model is derived from the structure of the underlying expert system. The limitation with this approach is that the novice student may not be employing a strict subset of the expert's skills, but using simplifications or other deviations. Therefore, one should not construct the student model just as an overlay of the final desired set of skills. To overcome this limitation Goldstein[23] employs an overlay approach to a *genetic graph* in which the student's knowledge is described in terms of the nodes of the graph, his learning behaviour in terms of the edges, and his progress in terms of the paths in the graph. The information represented at the nodes of the graph need not

be restricted to just the expert's skills or knowledge; rather, various modifications of these skills may be represented, including specializations, generalizations, or even erroneous procedures.

In their expert system BUGGY[6,7], Brown and Burton need to determine how a child's algorithmic behaviour differs from that of an expert—i.e. the correct use of various procedural skills. In this case it is clearly insufficient to represent the student's knowledge just as a subset of the expert's skills. One needs to represent misconceptions or mislearned subskills, which are not necessarily strict subsets of correct skills. In this case they use a *perturbation construct* to represent a student's misconceptions as variants of the correct procedural skills. We discuss this system in more detail later.

There are many different approaches to knowledge representation. Furthermore, there is no single standard approach to representing knowledge within the embedded domain expert or for modeling what the student knows or does not know. For excellent overviews of the different knowledge representation techniques that have been employed within ICAI systems see [3,44].

A third important feature of an ICAI system is the inclusion of some mechanism for tutoring or teaching. Ideally, the system should be able to make intelligent decisions regarding what, when, and how to teach. ICAI systems contain sufficient expertise to solve problems within a particular domain. Hence, they can employ this expertise to determine whether a student's answer is corrrect. However, they should go one step further; rather than just checking a student's answer as an end-product, the system should evaluate the process or procedure by which the student achieved the answer. In so doing, hopefully the system will determine where the student has gone astray and hence be able to determine which skills are lacking. The system should then provide help in the form of explanations, hints, and further examples. Ideally, the system should possess a variety of teaching strategies and be able to select the most appropriate for the student and task in question. A good teacher does not employ the same teaching methodology for all students or all lessons, and neither should a CAI system. The teaching strategies most commonly employed in ICAI systems are:

(i) coaching the student within a particular activity, such as a game-playing situation; the intent is to manipulate the environment and the coaching such that a particular skill(s) or general problem-solving ability is acquired;

(ii) questioning the student to encourage reasoning about current knowledge and to modify or formulate concepts; this may involve simulations or games in which the student can discover facts or laws;

(iii) providing tasks for the student and evaluating the responses in order to detect the student's misconceptions.

Unfortunately, most ICAI systems are limited in their knowledge of different teaching methodologies or strategies. One intriguing possibility is to provide the system with the capability to modify its teaching strategies based upon its success with a particular student or many students. In other words, the system should have the ability to learn from its experience. This is the intent of recent work by O'Shea regarding self-improving ICAI systems[36]. The interested reader should also consult the recent AI literature concerning learning (for example, see [32]).

Furthermore, it is a nontrivial task just to determine when to teach a particular skill. When teaching a complex sequence of skills or topics where prerequisite skills or knowledge may be critical to acquisition, it may be advantageous to incorporate planning strategies into the teaching module[41]. For an indication of what teaching strategies have been employed within ICAI systems, the interested reader should consult [3,44].

2.2 *Examples of ICAI systems*

ICAI systems have now been developed to teach a variety of subjects; examples of such systems include SCHOLAR[12], SOPHIE[8,9], BIP[2,53], WUMPUS[23,46], WHY[47], GUIDON[13,14,30], EXAMINER[35], ACE[42,45], QUADRATIC TUTOR[37,38], and EXCHECK[48]. It would be impossible to provide here a comprehensive survey of existing ICAI systems; instead, we briefly mention a few of these systems.

One of the first, and still better-known, ICAI systems is SCHOLAR[12], which was designed to teach facts about South American geography. Instead of storing geographic information in the form of prewritten frames (which is the approach employed in traditional CAI systems), the program was organized around an associated knowledge base that contained simple

geographic facts about industries, exports, populations, and capitals. SCHOLAR was designed to manipulate its knowledge base to generate factual questions, to evaluate the student's answers, and to answer the student's questions. Hence, in some sense, SCHOLAR "knew" the knowledge it was teaching—unlike previous traditional CAI systems, in which all questions, answers, and decisions had to be precoded. Furthermore, SCHOLAR is an early example of a *mixed-initiative* system; both the system and the student can initiate a dialogue by asking questions.

SOPHIE (SOPHisticated Instructional Environment)[8,9] is an ICAI system for electronic troubleshooting. Rather than instructing the user on the subject of electronics, SOPHIE provides the student with a learning environment in which to acquire problem-solving skills by trying out ideas. SOPHIE is a computer-based expert that helps the student to develop, test, and debug appropriate hypotheses.

The student is presented with a malfunctioning piece of electronic equipment. He must locate the faults by taking appropriate measurements. The student can ask questions about the measurements or which hypotheses should be considered. When the student forms a hypothesis, SOPHIE evaluates it and, if necessary, helps the student debug it.

Many of the early ICAI systems contained little expertise with regard to appropriate teaching methodologies, and the original version of SOPHIE was no exception. Early experiments with SOPHIE I and II indicated a need for additional coaching capabilities. SOPHIE has undergone several extensions, such as the inclusion of a troubleshooting game and a more sophisticated debugger and explainer, and currently contains some tools necessary for student modeling and coaching. However, Brown and Burton have concentrated their efforts regarding student modeling and tutoring/teaching strategies with somewhat simpler task domains[6,7,11], as we see later.

Another ICAI system that again demonstrated the need for both appropriate student models and teaching strategies is the WHY system, which tutors students about physical processes[47]. The system can carry on a simple dialogue regarding the causes of rainfall. The intent is to teach the student a "causal model" of the mechanisms underlying a wet climate. The original WHY system illustrates a common shortcoming of CAI systems; the system failed to diagnose the underlying misconceptions that are reflected in the student's errors.

After examining actual dialogues with tutors, Stevens and colleagues[47] concluded that much of a tutor's skill as a debugger (e.g. diagnosing conceptual bugs or misconceptions) depends upon knowledge about the types of conceptual bugs that students are likely to have, the manifestations of these bugs, and methods for correcting them. Consequently, an important component of any teaching system is a method for representing, diagnosing, and correcting misconceptions or bugs.

Other examples of ICAI systems arise in the context of game playing; examples include WEST[11] and WUSOR[23]. WEST is a computer-based coach for the mathematical game "How the West was Won." The intent is to provide sufficient guidance but still allow the child to learn by discovery. Simultaneously, one wants to detect the child's misconceptions; this is done through a *differential modeling* technique that compares what the student is doing to what the embedded expert expected. WEST also includes some general tutoring principles to help guide the coaching process.

WUSOR is a computer-based coach for the maze game WUMPUS. As in WEST, WUSOR contains expert knowledge about the game itself. This portion of the system can detect when the student's moves are nonoptimal and which skills are useful to discover the better alternatives. The tutoring component can then discuss the appropriate skills that have not yet been demonstrated by the player.

Unfortunately, all of the aforementioned systems are limited in their teaching ability. Current ICAI systems simply do not have the same level of flexibility or adaptability as a good human teacher. Furthermore, such systems are still restrictive in their use of natural language. SOPHIE has one of the more elaborate interfaces, which is capable of handling questions such as the following[9]:

What is the voltage at the base?
At the collector?
What about the current through the emitter?
Through R9?

Is that right?

What about Q5?

As illustrated by the above examples, SOPHIE can handle some forms of pronominal references, anaphorae, and ellipsis. SOPHIE's natural language interface is based upon a *semantic grammar*. In semantic grammars the possible statements to the system are characterized in terms of the underlying concepts of the domain. The main advantage of a limited domain such as SOHPIE's is that the number of possible activities within the domain is small; hence, there is a limited number of underlying concepts with which the system must deal.

3. LEARNING ENVIRONMENTS FOR STUDENT-INITIATED LEARNING

Most of the aforementioned ICAI systems are mixed-initiative systems, which allow either the student or the system to initiate a dialogue or pose questions. In some cases the system is acting as a teacher, but more often, the intent is for the system to be a coach to assist the student in discovering information or laws for himself.

This philosophy of discovery learning has been touted by many psychologists and educators (for example Piaget[4] and Bruner[10]). This is also part of the philosophical basis for Papert's programming language LOGO[39]). The intent is to create an environment that is conducive to spontaneous learning, the kind of learning by which a young child learns to talk or to walk. Papert developed LOGO in accordance with two fundamental heuristics of spontaneous learning. First, start from previous knowledge; a person is not able to make sense of a new experience and assimilate it unless it can be related to previous experience. Second, the learner should use the new ideas to 'make them his own': concepts are learned and remembered if they are important to the learner.

Unlike most CAI systems, which are interactive computer programs, LOGO is a programming environment. The child is the programmer; he/she manipulates the equipment, controls and builds the programming environment. In other words, LOGO is an attempt to provide an appropriate, friendly environment in which student-initiated learning (SIL) can take place.

A LOGO system generally includes a graphic 'turtle' that can be controlled by LOGO commands. Physical turtles, graphic sprites (entities of less power than the turtle, which can operate concurrently with it), music boxes, Meccano sets (including LOGO controlled cranes and lifts) have all been incorporated into various LOGO-based systems. By providing such tools and sufficient guidance the child can create programs, including simulations, for studying a variety of domains. LOGO has been used in a variety of projects, including making movies, composing tunes, writing concrete poetry, as well as investigating both simple and complex domains, such as elementary geometry, elementary motion, and balance problems in physics, differential geometry, and Newtonian physics[1].

LOGO has been used in a variety of settings with programmers ranging from preschool children to adults (for example, see [29,39,40,49]). It has also been successfully employed with children experiencing mathematical difficulties[27], adolescents with communication disorders[50,51], and mentally retarded or autistic children[22,52]. In the past couple of years LOGO's popularity has soared; there are now many LOGO-based courses and projects underway.

One very interesting recent project involved extending LOGO to include more elaborate objects, other than the original turtle, and to allow objects to interact. The extended system, LEPUS[20,21], is designed to increase LOGO's ability to represent complexity and to introduce a capacity for inconsistency. Consider Aesop's well-known fable regarding the tortoise and the hare. LOGO's turtle can easily be programmed to simulate the tortoise. Goforth, LEPUS's creator, wanted a means of representing the more complex, unpredictable hare.

LEPUS maintains the LOGO approach of providing the learner with an 'object-to-think-with' for exploring a new domain of knowledge. However, the 'object' is enriched with concepts for exploring the interaction of entities or systems over time. The differences involve a turtle with internal structure, a more complex environment including structured turtles, and a time variable to permit the environment and entities to evolve.

Of course, there is no reason for computer-based learning environments to be restricted to LOGO. In fact, both PROLOG and SMALLTALK have been used in similar situations. One particular project worth mentioning here is the recent work regarding *Programming by Re-*

hearsal[25]. The intent is to provide the user with a new method of programming in order to implement their own interactive, graphical, educational activities.

Using the REHEARSAL WORLD, which is an exploratory programming environment, nonprogrammers (such as teachers or children) can construct and modify interactive activities (called productions). To make a production the programmer employs predefined 'performers' that are placed on a 'stage' and 'auditioned.' The basic performers are buttons, numbers, texts, pictures, and turtles. Each performer has a set of predefined actions, each of which is associated with a specific 'cue.' In order to define the performers interactions (in terms of their predefined cues), some programming is required; however, most of the necessary code can be written by having the system watch as the designer rehearses the particular performers on the stage.

Gould and Finzer[25] envision a system in which the users are designers; after interacting with a production, the user will discover how it works and then be able to modify it for personal use. The REHEARSAL WORLD is currently implemented in SMALLTALK and runs on a Xerox Dorado. As an initial demonstration, Finzer[25] developed the *Symmetry Production*, which is an exploratory environment intended to provide the user with an understanding of rotational symmetry. The REHEARSAL WORLD also appears to be a useful introductory environment for SMALLTALK programmers.

4. EXPERT SYSTEMS FOR EDUCATIONAL DIAGNOSIS

As mentioned earlier, an important aspect of teaching is the ability to anticipate and diagnose a student's misconceptions. This means more than simply noting the child's errors; one must be able to determine the underlying cause of the errors. Some of the systems mentioned above make a concerted effort in this regard and attempt to maintain an accurate and current model of the student's knowledge, skills, errors, and misconceptions.

There is, however, another role for expert systems within the context of educational diagnosis. There are children within the regular school system who are experiencing severe learning difficulties. In order to plan an appropriate instructional program for a child with learning problems, the exact nature of the child's difficulties must be determined. This educational diagnosis should be initiated as soon as possible. Ideally, assessment should take place within the regular school environment by a teacher who is familiar to the child, preferably the regular classroom teacher or the resource-room teacher. Guidelines regarding appropriate procedures, differentiated teacher qualifications, and responsibilities are outlined in the SEECC (Standards for Educators of Exceptional Children in Canada) Model[19]. Unfortunately, educational assessment does not always follow the guidelines specified by the SEECC Model. Although most teachers can undertake initial assessment, many do not have sufficient expertise to independently undertake an in-depth diagnosis. It is therefore necessary to call upon the services of a local expert, such as a resource teacher, school psychologist, or outside agency, to augment the initial assessment with more sophisticated evaluation procedures, leading to advice, guidance, and prescriptive programming. In some cases, however, no initial screening is undertaken before the referral. Moreover, it can often involve a lengthy wait before the specialist is available.

The computer provides one means of facilitating educational diagnosis within the regular school environment. An expert system could be developed to guide the classroom teacher and/or resource person through the various stages of diagnosing learning disabilities from the initial screening to a prescription. At each step the expert system would analyze the available data and suggest an appropriate next step. It could request information regarding the child's developmental history or academic skills. The administration of a particular standardized test may be advised, or it might recommend further assessment of a skill or ability not within its domain of expertise. This might include consultation with a specialist or a referral to an outside agency.

The teacher or diagnostician would perform the required task, such as obtaining the requested data, or administering the appropriate test, and would supply this information to the system. After this new data had been assimilated and analyzed, the system would propose the next step, and so on. Eventually, the system would provide a summary of its diagnostic findings along with a prescription, including appropriate remedial activities and instructional techniques.

The system would not necessarily test the student directly, nor would it manage the testing activities. Rather it would guide the diagnostician. Interaction is between the computer and the

diagnostician, with the system posing questions or making appropriate suggestions. Interaction between the computer and the child could also be incorporated. In fact, administration and scoring of some standardized tests has been computerized[28,34]. Of course, computerized testing need not be restricted to standardized tests.

Whatever the area of assessment, the system must ultimately provide the user with a summary of the diagnostic findings and recommendations for remediation. The latter should include appropriate activities and instructional techniques, as well as suggestions regarding who should participate and where.

As an initial step in the development of a full expert system to guide a teacher through the various stages of diagnosing learning disabilities, we have designed and implemented a more limited expert system to assist in the assessment of reading problems[16,17,18]; the McLeod Educational Diagnostic Model[31] is employed as the underlying frame of reference. The present expert system guides the user through the various stages and levels of diagnosis—from the initial suspicion that a reading problem may exist, through to the point at which sufficient information has been gathered to plan an appropriate remedial program. Assessment begins with the gathering of relevant data concerning the child's physical, mental, emotional, social, and academic developmental history. In addition to the assessment of the child's general skills in academic areas, such as reading, spelling, and arithmetic, the expert system examines psycho-educational correlates that include those intellectual, visual, auditory, and language skill deficiencies that might be related to learning disabilities. As the assessment of the child's learning disabilities progresses, academic skills are subjected to finer and finer scrutiny until the nature of the child's problems has been pinpointed exactly.

For example, within the area of reading, one might assess the child's knowledge of sight words, decoding skills, phonics skills, reading comprehension, etc. Within each of these categories there is an enormous amount of information one might collect. For example, when reading aloud, what type of errors does the child make: substitutions, omissions, word reversals, etc? Does the child have more difficulty reading isolated words than short paragraphs, where context clues can be exploited? Does the child make more errors at the beginning of words, or are there particular word parts or endings that cause difficulties?

Just within the area of phonics there are many skills that could be assessed. For example, one should examine the child's ability to decode words containing single consonants or consonant blends at the beginning, in the middle, and at the end of words. Are there particular consonants or combinations that cause difficulties? Are there certain substitutions that the child makes? What vowels or vowel combinations create problems? There are many skills to assess; yet phonics is a small subset of the general area of reading.

Moreover, there are many related abilities which should be assessed, such as interpreting pictures, developing sentences or short stories, expressing one's own ideas or those presented in a given story. Because reading is not an isolated ability but a collection of many interrelated skills, assessing a child's reading can be a long, arduous task. Hence, part of the expert's job is to determine what to assess in each individual case and to perform an efficient diagnosis with regard to both the expert's and the child's time.

In addition to determining precisely which skills are missing or inadequate, the expert system must ascertain which skills or abilities have been mastered and which areas represent relative strengths for the child. Such information is necessary for the development of an appropriate remedial program. It is important to know whether a certain skill is a relative strength or weakness for a particular child (e.g. in comparison to the child's other skills), as well as if the skill is a strength or weakness (e.g. in comparison to the child's chronological peers). For example, perhaps the child's knowledge of sight words is considered weak in contrast to other children of the same age, but in comparison to his/her decoding skills, the child's ability to read sight words may be a relative strength, and, hence, should be exploited in the remediation process.

The current expert system is implemented as a production system and is programmed in LISP; for further information regarding production systems, the various components of an expert system, and appropriate authoring tools, the reader should consult [26]. In a production system the rules represent the distillation of the human expert's knowledge: i.e. an encoding of how the experts handle particular situations. Hence, in this case, information regarding diagnostic procedures is encoded into the system's production rules. The development of the rules is, in

itself, a major undertaking. For each component of the diagnostic model, one must ascertain:

(i) what data are normally collected;

(ii) the usual sources of such data, e.g. questionnaires, tests, previous assessments;

(iii) how this information is applied, e.g. what facts, suspicions, hypotheses emerge when this new information is assimilated.

To establish this information, diagnoses previously undertaken at the Institute of Child Guidance and Development at the University of Saskatchewan were examined.

The system's performance has been evaluated by comparing its diagnostic findings to those of human diagnosticians. In general, the results of the comparison are encouraging. Not only are the system's diagnoses accurate, but because the expert system can perform appropriate analyses (for example, of error patterns) more quickly and more accurately than most diagnosticians, the system's diagnoses tend to be more extensive (within certain limited areas of expertise) than those of human diagnosticians. For example, within the area of phonics, the expert system performs more analysis of error patterns than most diagnosticians have the time to do.

Ultimately, one would like an expert system(s) to assist teachers in diagnosing and assessing learning difficulties in a variety of areas, such as language, reading, spelling, mathematics, etc. Consider the assessment of a child's arithmetic skills. Having determined that this is an area of weakness and, perhaps, even that subtraction is a problem, one still must determine what aspects of subtraction are proving difficult. Is it poor knowledge of number facts? Is it that the child does not borrow correctly? To ascertain the precise nature of the child's subtraction difficulties, the diagnostician would generally undertake some informal (e.g. nonstandardized) testing. Much of this stage of assessment can be computerized, as nicely demonstrated by the recent BUGGY system of Brown and Burton[6,7].

BUGGY[6,7] is a program that can determine a student's arithmetic misconceptions or bugs. The system is based on the belief that a student's algorithmic errors are not random, but are consistent, discrete modifications of the correct arithmetic procedure. BUGGY attempts to determine what internalized set of incorrect instructions or rules gives results equal to the student's answer. In other words, given a new problem, BUGGY should be able to predict the student's response.

Included in BUGGY's domain expert is information regarding common arithmetic bugs or procedural errors. In the case of subtraction, 110 primitive bugs are included as well as 20 common compound bugs. The results of all 130 bugs are compared with the student's answers. Based upon these comparisons, the system selects a subset of bugs, each of which explains at least one of the student's wrong answers. The elements of this initial set of bugs are then combined to generate additional hypotheses for the particular student in question. Now the system starts eliminating hypothesized/proposed bugs; for example, one rule employed in this process is to remove bugs that are subsumed by other primitive bugs. This is basically an iterative procedure of removing subsumed bugs and forming combinations of the remaining bugs. Ultimately, each of the remaining proposed bugs is classified according to how well it explains the student's answers. This classification procedure takes into account the number of predicted correct and incorrect answers as well as the number and type of mispredictions. Hopefully, at the end of this classification procedure, one bug can be selected as the best explanation of the student's erroneous responses; this bug may be one of the original primitive or compound bugs or a combination thereof.

BUGGY has been successfully employed within the regular classroom and has been used with more than a thousand students. The type of testing and analysis of errors performed by BUGGY is often a time-consuming task for a teacher or diagnostician. It is this type of individual informal testing that is necessary during the latter stages of educational diagnosis in order to determine the exact nature of a student's difficulties. A system such as BUGGY could be incorporated into a more comprehensive system for diagnosing learning disabilities, as described above. In fact, such expert systems could be developed to guide, administer, and analyze informal testing for a variety of areas such as phonics, spelling, and other aspects of mathematics. The expert system described in [16] is already capable of a substantial amount of analysis concerning a child's phonics skills. Another example of an appropriate diagnostic system within a mathematical domain is Sleeman's system for detecting a student's bugs or misconceptions

within the area of algebra[43]. The result of incorporating such specialized expert systems into the latter stages of an expert system that guides the diagnostic process would be a powerful tool for facilitating educational diagnosis and assessment within the regular classroom environment.

5. CONCLUSIONS

Although the systems discussed herein still fall short of what we would like to see placed within the regular school environment, they demonstrate the impact that recent AI research is having within the field of education. For example, current ICAI systems are a vast improvement over the traditional frame-based CAI systems. Advances made within the central AI research areas, such as knowledge representation, natural language understanding, reasoning/inferencing, and learning/discovery, will undoubtedly continue to be reflected within the educational field, particularly in instructional and diagnostic systems.

REFERENCES

1. H. Abelson and A. diSessa, *Turtle Geometry*. MIT Press (1981).
2. A. Barr, M. Beard and R. C. Atkinson, A rationale and description of a CAI program to teach the BASIC programming language. *Instructional Sci.* **4**, 1–31 (1975).
3. A. Barr and E. A. Feigenbaum (Editors), *The Handbook of Artificial Intelligence*, Vol. 2. William Kaufmann (1982).
4. M. Boden, *Piaget*. Harvester Press (1979).
5. J. S. Brown, Uses of artificial intelligence and advanced computer technology in education, in *Computers and Communications: Implications for Education* (Edited by R. J. Seidel and M. Rubin), pp. 253–270. Academic Press (1977).
6. J. S. Brown and R. R. Burton, Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Sci.* **2**, 155–192 (1978).
7. J. S. Brown and R. R. Burton, Diagnosing bugs in a simple procedural skill, in *Intelligent Tutoring Systems* (Edited by D. Sleeman and J. S. Brown), pp. 157–183. Academic Press (1982).
8. J. S. Brown, R. R. Burton and A. G. Bell, *SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting (An Example of AI in CAI)*. Bolt, Beranek and Newman, Inc., Report 2790 (1974).
9. J. S. Brown, R. R. Burton and J. de Kleer, Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III, in *Intelligent Tutoring Systems* (Edited by D. Sleeman and J. S. Brown), pp. 227–282. Academic Press, New York (1982).
10. J. Bruner, *The Process of Education*. Harvard University Press (1977).
11. R. R. Burton and J. S. Brown, An investigation of computer coaching for informal learning activities, in *Intelligent Tutoring Systems* (Edited by D. Sleeman and J. S. Brown), pp. 79–98. Academic Press, New York (1982).
12. J. Carbonell, AI in CAI: an artificial-intelligence approach to computer-assisted instruction. *IEEE Trans. Man-Machine Syst.* **MMS-11**, 190–202 (1970).
13. W. J. Clancey, *Transfer of Rule-based Expertise Through a Tutorial Dialogue*. Computer Science Dept., Stanford University, Report No. STAN-CS-769 (1979).
14. W. J. Clancey, Tutoring rules for guiding a case method dialogue, in *Intelligent Tutoring Systems* (Edited by D. Sleeman and J. S. Brown), pp. 201–225. Academic Press, New York (1982).
15. W. J. Clancey and R. Letsinger, NEOMYCIN: reconfiguring a rule-based expert system for application to teaching. *Proc. 7th IJCAI*, pp. 829–836 (1981).
16. M. J. Colbourn, *Computer-Guided Diagnosis of Learning Disabilities: A Prototype*. M. Ed. Thesis, Dept. for the Education of Exceptional Children, University of Saskatchewan (1982); also Dept. of Computational Science, University of Saskatchewan, Technical Report No. 82-8; also ERIC Report No. ED 222-032.
17. M. J. Colbourn and J. McLeod, The potential and feasibility of an expert system for educational diagnosis. *Proc. IFIP*, pp. 891–896 (1983).
18. M. J. Colbourn and J. McLeod, Computer-guided educational diagnosis: a prototype expert system. *J. Special Ed. Tech.* **6**, 30–39 (1984).
19. Committee on Teacher Education and Professional Standards, *Standards for Educators of Exceptional Children in Canada*. The National Institute for Mental Retardation (1971).
20. D. Goforth, *LEPUS: An Object Oriented Extension of Turtle Geometry*. M. Sc. Thesis, Dept. of Computational Science, University of Saskatchewan (1983); also Technical Report 83-9.
21. D. Goforth and G. McCalla, LEPUS: a language to support student initiative learning in non-mathematical domains. (to appear in *AEDS J.*).
22. E. P. Goldenberg, *Special Technology for Special Children*. University Park Press (1979).
23. I. P. Goldstein, The genetic graph: a representation for the evolution of procedural knowledge, in *Intelligent Tutoring Systems* (Edited by D. Sleeman and J. S. Brown), pp. 51–77. Academic Press, New York (1982).
24. I. P. Goldstein and B. Carr, The computer as a coach: an athletic paradigm for intellectual education. *Proc. 1977 Annual ACM Conf.*, pp. 227–233 (1977).
25. L. Gould and W. Finzer, *Programming by Rehearsal*. Xereox PARC, Report SCL-84-1 (1984); a short version of this paper also appears in *BYTE Mag* **9**(6) (1984).
26. F. Hayes-Roth, D. A. Waterman and D. B. Lenat (Editors), *Building Expert Systems*. Addison-Wesley, New York (1983).
27. J. Howe and T. O'Shea, Learning mathematics through LOGO. *ACM Sigcue Bull.* **12**, 2–11 (1978).

28. J. H. Johnson and T. Williams, Clinical testing and assessment: using a microcomputer for on-line psychiatric assessment. *Behavior Res. Meth. Instrument.* **10**, 576–578 (1978).
29. Lamplighter School, Learning with LOGO at the lamplighter school. *Microcomputing*, 42–47 (Sept. 1981).
30. B. London and W. J. Clancey, *Plan Recognition Strategies in Student Modeling: Prediction and Description*. Dept. of Computer Science, Stanford University, Report No. STAN-CS-82-909 (1982).
31. J. McLeod, Institute of child guidance and development at the University of Saskatchewan. *J. Learn. Disabil.* **15**, 290–293 (1982).
32. R. S. Michalski, J. G. Carbonell and T. M. Mitchell, *Machine Learning*. Tioga Publishing (1983).
33. D. Michie (Editor), *Expert Systems in the Micro-Electronic Age*. Edinburgh University Press (1979).
34. M. P. Nichols and I. J. Knopf, Refining computerized test interpretations: an in-depth approach. *J. Personality Assess.* **41**, 157–159 (1977).
35. C. E. Oleson, EXAMINER: a system using contextual knowledge for analysis of diagnostic behavior. *Proc. 5th IJCAI*, pp. 814–818 (1977).
36. T. O'Shea, *Self-Improving Teaching Systems*. Birkhauser-Verlag (1979).
37. T. O'Shea, Rule-based computer tutors, in *Expert Systems in the Micro-Electronic Age* (Edited by D. Michie), pp. 226–232 (1979).
38. T. O'Shea, A self-improving quadratic tutor, in *Intelligent Tutoring Systems* (Edited by D. Sleeman and J. S. Brown), pp. 309–336. Academic Press, New York (1982).
39. S. Papert, *Mindstorms*, Basic Books (1980).
40. S. Papert, D. Watt, A. diSessa and S. Weir, *Final Report on the Brookline LOGO Project: Part II*: Project Summary and Data Analysis. AI Lab, Massachusetts Institute of Technology, AI Memo No. 545 (1979).
41. D. R. Peachey, *An Architecture for Plan-Based Computer Assisted Instruction*. M. Sc. Thesis, Dept. of Computational Science, University of Saskatchewan (1983); also Technical Report 83–11.
42. D. Sleeman, A system which allows students to explore algorithms. *Proc. 5th IJCAI*, pp. 780–786 (1977).
43. D. Sleeman, Assessing aspects of competence in basic algebra, in *Intelligent Tutoring Systems* (Edited by D. Sleeman and J. S. Brown), pp. 185–199. Academic Press, New York (1982).
44. D. Sleeman and J. S. Brown (Editors), *Intelligent Tutoring Systems*. Academic Press, New York (1982).
45. D. Sleeman and R. J. Hendley, ACE: a system which analyses complex explanations, in *Intelligent Tutoring Systems* (Edited by D. Sleeman and J. S. Brown), pp. 99–118. Academic Press, New York (1982).
46. J. L. Stansfield, B. P. Carr and I. P. Goldstein, *WUMPUS Advisor I: A First Implementation of a Program that Tutors Logical and Probabilistic Reasoning Skills*. AI Laboratory, Massachusetts Institute of Technology, Memo No. 381 (1976).
47. A Stevens, A. Collins and S. E. Goldin, Misconceptions in students' understanding, in *Intelligent Tutoring Systems* (Edited by D. Sleeman and J. S. Brown), pp. 13–24. Academic Press, New York (1984).
48. P. Suppes (Editor), *University-Level Computer Assisted Instruction at Stanford*: 1968–1980. Institute for Mathematical Studies in the Social Sciences, Stanford University (1981).
49. D. Watt, *Final Report on the Brookline LOGO Project: Part III: Profiles of Individual Student's Work*. AI Lab. Massachusetts Institute of Technology, AI Memo No. 546 (1979).
50. S. Weir, LOGO as an information prosthetic for communication and control. *Proc. 7th IJCAI*, pp. 970–974 (1981).
51. S. Weir, LOGO: a learning environment for the severely handicapped. *J. Special Ed. Tech.* **5**, 20–22 (1982).
52. S. Weir and R. Emanuel, *Using LOGO to Catalyse Communication in an Autistic Child*. Dept. of Artificial Intelligence, University of Edinburgh, DAI Research Report No. 15 (1976).
53. K. Westcourt, M. Beard and L. Gould, Knowledge-based adaptive curriculum sequencing for CAI: application of a network representation. *Proc. 1977 Annual ACM Conf.*, pp. 234–240 (1977).