Computers Educ. Vol. 20, No. 1, pp. 27–43, 1993
Printed in Great Britain. All rights reserved

0360-1315/93 $6.00 + 0.00
Copyright © 1993 Pergamon Press Ltd

AN APPROACH TO DEVELOPING INTELLIGENT TUTORS IN MATHEMATICS

HYACINTH S. NWANA
Department of Computer Science, University of Keele, Keele, Staffordshire ST5 5BG, England

Abstract—Mathematics is highly structured and underpins most of science and engineering. For this reason it has proved a very suitable test domain for intelligent tutoring system (ITS) research with the result that probably more tutoring systems have been structured for the domain than for any other. However, there still exists no consensus on any approach for the design of such systems. Consequently, existing ITSs in the domain suffer from a considerable number of shortcomings which render them 'unintelligent'.

This paper after rigorously examining the shortcomings of existing approaches, presents an alternative approach to constructing ITSs in mathematics which has been demonstrated to have improved on at least some of the shortcomings of existing approaches. It also provides a flavour of how this approach has been implemented in the FITS system.

INTRODUCTION

Current approaches to the construction of intelligent tutoring systems (ITSs) in mathematics can be grouped under two main headings:

(1) The mal-rule approach
(2) The model-tracing approach.

This paper critiques both these approaches and lists some of their example systems. In the light of these analyses and others, another approach designed to produce ITSs to improve on at least some of the shortcomings of the above-mentioned is presented.

THE MAL-RULE APPROACH

Mal-rules (or bugs) represent the student's misunderstandings of the domain. They are central to this approach. Systems based on mal-rules have the characteristic of being diagnostic in nature: they end up diagnosing the student's misconception or give up. According to West, the rationales for the approach are two-fold: firstly, that "there is hardly a skill in the teacher's repertoire that is more important than the ability to identify pupils' errors and to prescribe appropriate remedial procedures", and secondly, that errors may be "springboards" for students to understand mathematics (quoted in [1]).

However, a more accurate distinction between this and the second approach mentioned above is made by answering Vanlehn's[2] bandwidth question: how much of the learner's activity is available to the diagnostic program? Most ITSs in mathematics work on the low end of the information band where only the final state is available to the system, i.e. only the student's answer to a question is available to the system [3]. Such tutors are henceforth referred to in this paper as mal-rule tutors. Sleeman [4] notes that implementors of such ITSs frequently perform the following steps:

(1) Analyse protocols from students in the target population solving typical tests and codify their difficulties/misunderstandings. (This may sometimes involve detailed 'clinical' interviews with students.)
(2) Create databases which include codings of the mal-rules (bugs) observed in Step (1).
(3) Implement and use the ITS with students and in particular note student errors which are *not* spotted by the system.

(4) Carry out detailed interviews to determine the nature of these errors and code them as additional mal-rules. Steps (3) and (4) are repeated until the system captures the majority of the bugs which occur with the target population.

Some examples of mal-rule ITSs in mathematics

Table 1 lists examples of mal-rule ITSs in mathematics that have been developed to date. It is reasonably representative. It is also important to highlight the fact that most of these were developed as prototypes and largely remain so. Hence, hardly any have been tested on more than a few people. These ITSs do a lot less than the domain indicated suggests.

Critique of the mal-rule approach

The concept of 'debugging' is the major device used in most mal-rule systems to achieve the diagnosis of errors; DEBUGGY is perhaps the most obvious example. Some of the criticisms levelled at such systems are [13–15]:

(1) The knowledge involved in debugging algorithms reveals rich interconnections between different domains of mathematics. This restricts the value of single domain systems.

(2) Some researchers have argued that debugging is an intellectual task which might have great educational value [15] and hence pupils should be major agents of the debugging process, since they are the ones who have most to gain from it.

(3) There are a variety of idiosyncratic methods and approaches that can be used successfully to solve most mathematical problems. Systems such as DEBUGGY neither know about such methods nor can they learn them, because they have no access to how pupils solve problems in the first place. Thus if a range of pupil solutions to subtraction problems is considered, DEBUGGY fares no better than a human teacher, since it will completely fail to debug errors in idiosyncratic methods[15].

(4) Many systems seem to support a split between an algorithm and its applications. Divorcing these two, it is argued in [15], "is a device of very doubtful pedagogical value. Subtraction should not be viewed as a self-contained topic which is separate from 'mathematical thinking'. It should be viewed as an integral part of mathematics and mathematical thinking and teaching should focus on extending and strengthening links and other areas of mathematics". O'Shea *et al.* [16] argue that it is this very divorce that brings about the systematic errors that children make in mathematics.

(5) There is still a very high percentage of unexplained errors ($\sim$30–40%)[16].

(6) This approach presumes that the novice (student) deviates from the expert (tutor) in some simple way. This is a very common misconception. For example, the student might be viewed as having too few routines, and needs to add more to his/her repertoire; or may be viewed as having some bugs or mal-rules. It is argued in [15] that both these simplistic views are wrong and that there are major differences between novices and experts that are not so readily remedied.

(7) The premise that students' errors are systematic is a basic one, but the nature of the systematicity is difficult to define. This leads to the question of bug stability. Are children's errors transient in nature? Children can be seen to employ two different buggy procedures in solving almost identical problems within a single session [17]. Therefore, classifying student's behaviour on some single occasion as being due to various types of buggy behaviour may be woefully misleading.

Table 1. Some examples of mal-rule ITSs in mathematics

BUGGY[5]	Diagnosis in basic subtraction
DEBUGGY[6]	Diagnosis in basic subtraction
IDEBUGGY[6]	Diagnosis in basic subtraction
ATDSE[7]	Diagnosis in basic subtraction
EDSMB[8]	Diagnosis in basic multiplication
HELPERR[9]	Diagnosis in basic addition
LMS[10]	Diagnosis in basic algebra
PIXIE[11]	Diagnosis/tutoring in basic algebra
WEST[12]	Coaching in basic arithmetic

THE MODEL-TRACING APPROACH

Model-tracing approaches have been around ~~for several years now~~ with its foremost protagonist being the Carnegie Mellon psychologist John Anderson. However, other researchers had earlier suggested such approaches. The technique (model-tracing) is so termed to express the fact that the student is made to follow the system's model quite closely[18]. These tutors guide students through a problem trying to make their steps correspond to those of the ideal student model.

Model-tracing tutors use the expertise of their problem solver to predict the steps the students will take while working on a problem. The problem solver generates all the possible next steps, both correct and incorrect according to the database of rules. They are compared to the student's step and the rule that matches is selected as an interpretation of the actions. If the tutor cannot provide a model-driven interpretation of the student's steps, the tutor signals that it does not understand the last input and after a pre-determined number of trials, it simply suggests the next best step according to the ideal model.

The main difference between model-tracing tutors and mal-rule tutors lies in the answer to Vanlehn's[2] bandwidth question noted earlier. These ITSs have access to intermediate states: hence, they work with more information (larger bandwidth) than mal-rule tutors. Because the diagnostic module needs as much reliable information as possible about the learner's mental state, bandwidth is critical in designing ITSs. However, the main features of model-tracing are the following[19, 20]:

(1) The tutor constantly monitors the student's problem solving and provides direction wherever the student wanders off path.

(2) The tutor tries to provide help with both the overt parts of the problem solution and the planning. However to address the planning, a mechanism had to be introduced in the interface (in this case menus) to allow the student to communicate the steps of the planning.

(3) The interface tries to eliminate aspects like syntax checking which are irrelevant to the problem solving skill being tutored.

(4) The interface is highly reactive in that it does make some response to every symbol the student enters.

Some examples of model-tracing ITSs in mathematics

GEOMETRY Tutor[21,22]	Geometry proofs tool
ALGEBRA Tutor[22]	Algebraic proof tools
INTEGRATION Tutor[23]	Basic integral calculus
ALGEBRALAND[24]	Algebraic proofs tool

Critique of the model-tracing approach

There appear to be several 'brands' of model-tracing, e.g. the discovery-learning type as exemplified in ALGEBRALAND or the more rigid approach in the GEOMETRY tutor. Anyway, as previously noted, Anderson's[18] is by far the most popular. This is supported by recent attempts at using model-tracing approaches which have produced systems (intermediate systems) more towards the Andersonian types, e.g. [25]. Furthermore, the work of Anderson's team is also amongst the most principled and documented of current ITS research. As a consequence, in listing some of the shortcomings of this approach, there is an obvious bias towards his 'brand' of model-tracing. Model-tracing tutors' drawbacks include:

(1) "They seem to incorporate a very dogmatic and authoritarian approach to education"[26]. This is because the main driving force behind such tutors is the detection of deviation from the ideal student model (expert problem solver). This has obvious dangers in that the system will reject any other correct approach to solving the problem if it differs from the path in the system. The model-tracing approach currently keeps the student on one of the correct solution paths in the problem; normally the optimal path[27]. It is a common misconception to talk about the 'correct' way an expert solves the problem. Ridgway[15] notes that a major characteristic of experts is that they can solve the same problem in a variety of ways. In short, such tutors are incapable of supporting the variety of idiosyncratic methods that can be used to solve most problems.

(2) They feign omniscience[28]. But the student could possess strategies that are equally as good or even better than the 'hard-wired' strategy in the ideal student model. Many researchers have called for a more collaborative approach, e.g.[29]. However, it is not always clear that the mechanisms for implementing this have been available.

(3) It is argued in[30] that ITSs often have the advantage of an explicit set of tutorial strategies. In model-tracing ITSs, these strategies are often 'hard-wired' in these tutors. Therefore, there is really no explicit representation of knowledge, i.e. the domain-independent and the domain-dependent parts are not clearly delineated. This also leaves no room for improvement by the system itself, though the progress of the student does provide feedback on the teaching process. In brief, such systems cannot be easily adapted to tutor another domain [20]: to do so basically requires a total reconstruction of the new system; neither can they learn or improve their strategies over time. There are thus undisputably more benefits to be reaped by constructing systems which have their knowledge explicitly represented. To be fair, this issue is non-trivial: in fact, knowledge representation still remains one of AI's unsolved problems. However, some researchers have had some success, albeit limited, at building self-improving tutors [23, 31], largely as a result of explicit knowledge representation.

(4) The important educational activity of debugging is taken away by such tutors since this approach, in principle, does not allow for floundering. It may be therefore deemed by some students as being very intrusive and definitely of limited value at teaching debugging skills, i.e. such systems may destroy the student's personal motivation or sense of discovery. This limitation also applies to the mal-rule approach. This is the central criticism that Papert and his supporters rightly level against ITSs. They claim the approach they champion fully addresses this problem, but there are still many questions about their LOGO technique that requires answering. Indeed, this is a difficult problem; tutoring involves so many subtle skills, e.g. when is it the right time to interrupt?, which good human teachers are gifted with. It is still a major research endeavour how to articulate the knowledge behind such subtle tutoring decisions into ITSs.

(5) The style of teaching never changes in such tutors. This is because the rules and mal-rules have merely been 'hardwired' into the system and such designs generally tend to lead to inflexible systems[32]. Many researchers, e.g.[33], have stressed the importance of a more flexible style of tutoring, though they seldom indicate how to go about implementing this.

(6) The success of such tutors depends heavily on the number of correct rules in the ideal student model as well as the number of mal-rules in the bug catalogue. For instance, the LISP tutor now contains more than 1200 rules [20], more than half of which are mal-rules [19]. Initially, it had 325 production rules for planning and encoding LISP programs and 475 faulty versions[20]. There are so many rules because of the attempt to encode all the possible paths the student could possibly take. These rules are obtained after painstakingly analysing numerous problem-solving protocols and considerable theoretical analysis of the problem solving domain. One questions how much time and effort ought to go into this analysis phase considering that a handful of mal-rules tend to account for a large percentage of the errors in the domain. Also, the time spent by such systems in diagnosing students' misconceptions could certainly be reduced if students were made to specify what they intended to do next. The uncertainty in what path a student would take is then reduced. Efficiency would also be enhanced with increased flexibility.

(7) The main limitations of earlier CAI systems were that they provided neither feedback nor individualization. Systems such as the Geometry tutor definitely do provide feedback mostly in the form of interrupting either to give advice or for diagnosis when the student wanders off the 'correct' path. However, the system lacks adequate individualization. This is probably the most severe of all the limitations of the model-tracing approach, at least as exemplified in systems like the LISP and Geometry tutors. This approach neither supports nor maintains any student models or student history files. Consequently, these tutors do not learn in any way about the student as human tutors do. Therefore such systems would not tutor a weak student any differently from another weak student though their levels of understanding might be significantly different. Self[28] strongly argues in favour of the rehabilitation of student models in intelligent tutoring systems.

(8) Some of these tutors tend to do too much for the student. For instance, in ALGEBRALAND, students do not even need to type in anything on the keyboard. It performs all the manual calculations for the student. In fact, in most cases, a response can be generated by pointing to parts

of the existing expressions in the current equation window using the mouse; the idea being to concentrate only on aspects that are relevant to the problem-solving skill(s) being tutored. This can be viewed as an advantage or disadvantage depending largely on what view of tomorrow's education one holds. If one is of the radical view that the classrooms of today have no place in tomorrow's schools[14], then this is certainly an advantage. If one rather holds the conventionalist view that present classrooms are here to stay, then many would argue that it is a disadvantage. As[27] rightly point out, it raises the 'training wheels' issue; what happens when the student leaves such a helpful environment that supports problem solving and has to face the realities of today's pencil-and-paper? It is certainly true that many errors would otherwise occur through slips and mis-strikes of keys or mis-copying of expressions but such or equivalent errors also in reality when using pencil and paper and students must learn how to recover from them. Admittedly, such help-systems would be very useful in the very early stages of learning a problem-solving skill when otherwise the student may be frustrated into giving up due to irrelevant errors. Perhaps they should therefore be used only at this very early stage and then the student should proceed to more lifelike systems. In summary, it seems that there is a good case for constructing systems which are closer to the reality of the day.

These are only the main criticisms levelled against the model-tracing approach. In fact, some of the shortcomings of the mal-rule approach also apply to this model-tracing one.

The effect of many of these limitations is to produce a relatively inflexible tutor as the systems strive to anticipate all the possible steps the student might make. Therefore, from the ITS perspective, the model-tracing approach, as exemplified by Andersonian tutors, tends to produce tutors somewhat towards the CAI/CAL end of the spectrum in spite of their apparent cognitive foundation. Wenger[34] enforces this view as he notes that "... the notion of a compiled ITS corroborates the CAI flavour of this approach to knowledge communication". Anderson and his colleagues have acknowledged the inflexibility in style of this approach and their recent efforts are enhancing the paradigm's flexibility by proposing a new production system, PUPS, to replace GRAPES on which the current tutors are based[20]. PUPS is meant to have improved on some of the shortcomings of GRAPES, but Anderson and his colleagues have not yet based any ITS on it.

In effect, the two predominant approaches to developing ITSs in mathematics have been observed to have numerous shortcomings; this contributes to the lack of consensus on any approach for the design of such systems. The next section proposes a novel approach which has been demonstrated to have improved on at least some of the shortcomings of the two main existing ones.

THE APPROACH

The shortcomings of existing approaches listed in this paper, the conclusions which were drawn from empirical work with the reasonably complex domain of fractions[35,36], the conclusions drawn from a bug theoretical study[37,38] and previous work of other researchers, e.g.[13], all contributed to the novel approach to be described. They not only provide ideas on how to approach building an ITS for the fractions domain but for other mathematical as well as suitably structured ones. The approach like the other two consists of a list of features (principles) which should be abided by, along with some justifications where deemed appropriate. In effect, mathematical ITSs should be capable of the following:

(1) *Pre-model the student.* In our bug theoretical work, it was observed that some errors children make result from deficient knowledge. "The major problems in automated tutoring is that the system needs to know what the student knows and understands, what his misconceptions are"[33]. Pre-modelling should identify these prerequisite skills, concepts and facts that are or are not possessed by the student in question as well as coverage of the syllabus. If the student does not exhibit enough prerequisite knowledge, then the necessary prerequisite should be presented[39]. Besides, it also seems unreasonable to assume that all learners start off with the same structuring of space[40].

(2) *Pre-teaching prerequisite skills.* Most of mathematics is considered hierarchical in nature, that is certain skills are prerequisites to higher level skills[41]. As mentioned earlier, the empirical study

clearly showed that errors resulted due to deficient mastery of prerequisite skills, facts and concepts. It is inevitable that if students do not possess the prerequisite skills, they would encounter difficulties with the goal topic. For instance, it is pointed out in[42] that if a student consistently solves tasks such as $2*3x + 4x = 22$ as $x = 22/14$, then tutors should recognize his consistency in making precedence errors and probably switch focus from algebra to arithmetic precedence until the student shows signs of understanding precedence in arithmetic, and then resume algebra tutoring. Most importantly, this will also strengthen links with other areas of mathematics. It is rightly pointed out in[15] that current systems, e.g. BUGGY/Geometry Tutor, support a split between algorithm and applications as they view topics they tutor as separate from 'mathematical thinking'. The ITS should then be able to teach or reteach the necessary prerequisites prior to the teaching of the goal skills. Another likely advantage is that pre-teaching would also make the diagnostics or any information generated by the tutor more meaningful, as the student would better understand what to do.

(3) *Provide examples (sequence of examples/lessons) such that Vanlehn's felicity conditions are satisfied.* The bug theoretical study highlighted that, so often, teachers or books assume too much of the students in their examples as well as their lessons. This view is also supported in[32]. Vanlehn's[43] felicity conditions address this issue.

(4) *Monitoring the student step by step.* The case for monitoring (or model-tracing) was clearly demonstrated in[36,37]. This view is enforced by[44] that "... the ideal educational system will carefully monitor interaction with the student ...". Besides, some systems, e.g. PIXIE, which are based on the mal-rule approach have been enhanced to make use of intermediate steps[42], albeit in very restricted fashions, to discriminate amongst possible models—a step towards a more monitoring-type approach. In summary, the system should then be able to monitor and comment upon the student's actions, recognize optimal, less than optimal, and clearly irrelevant actions. It should also offer help, hints, explanations and tutoring advice as appropriate[45].

(5) *Diagnose in a problem-solving context.* Diagnosis becomes easier and more efficient as the system has access to intermediate steps. This is because the problems of uncertainty and/or combinatorial explosion faced by DEBUGGY and PIXIE when carrying out diagnoses (assigning credit/blame) are substantially reduced. Self[28] also stresses that ITSs should avoid guessing and that they should get the students to tell them what they need to know. This point is well expressed in[40]: "an ICAI system should endeavour to interpret students' inputs, not merely in terms of knowledge understood or not, but as evidence about the student's goals, i.e. what he is trying to achieve". Diagnosis will be even further enhanced if this is done, for example, if the student specifies what operation will be carried out.

(6) *Be capable of getting the student to communicate intensions (plans) prior to executing them.* This follows from the previous feature.

(7) *Remediate in a problem-solving context.* The empirical study observed that systems of the "buggy" type print the inferred bug. This has been shown to be of limited value as there is no point diagnosing if one cannot remediate[28]. Remediation would also be easier and of more use to the student since the system would be monitoring every intermediate step. It should take the form of diagnosing the error, providing the correct answer, instructing the student what to do next or giving hints[46].

(8) *Keep information about the student's ability and progress (model the user).* Student information databases, such as student models or student history files, are conspicuously absent from model-tracing tutors, at least as exemplified in the Geometry and LISP tutors[21]. However most researchers seem to agree that to provide truly individualized instruction, which ITSs strive or should strive to do, such information is vital[28,40,47,48]. In brief, the review clearly notes that the student model is a key component of an ITS.

(9) *Be capable of testing the student's understanding.* This might be the student's understanding of prerequisite skills, e.g. as in feature (1), or understanding of the goal skill. The results of the testing exercise should be used to modify the student model.

(10) *Be capable of supporting various idiosyncratic ways which the student might choose to solve the problem.* This is contrasted with the ideal problem solver of the LISP and Geometry tutors which are 'hardwired' with the 'correct' way of solving the problem: or mal-rule tutors, which only normally request the final answer. Nonetheless, the ITS should be capable of comparing solutions

and indicating to the student if the correct solution is sub-optimal and possibly present the optimal solution. It must also be capable of solving the problems it presents to the students[49].

(11) *Have explicit representations of the knowledge of the student, knowledge of tutoring strategies, knowledge to be taught, the curriculum knowledge and the more enduring characteristics to which instruction should be sensitive.* CAI's main weaknesses stem from its implicit representation of knowledge. It has been argued that a system would be more 'intelligent' if such knowledge was all made explicit. Woolf[49] dwells on this premise.

(12) *Motivate and support a more flexible style of tutoring.* The empirical study pointed out that factors such as boredom and motivation are involved in learning[50]. Suppes[51] notes that the absence of sustained work on motivation in ITS research is its most serious omission. However, it is still extremely difficult to detect or model such phenomenological factors. As a safeguard, ITSs should be as motivating as possible to reduce the possibility of the student getting bored. 'Buggy' tutors were observed to be very boring and uninteresting in their style of tutoring. Model-tracing tutors were also observed to be inflexible. They should be more flexible by playing a more collaborative/mentor role for the weaker/better students respectively[33].

(13) *Maintain control over the whole tutoring endeavour and support a more mixed-initiative interaction.* They should be simple to use, able to detect and report minor/syntactic errors and maintain a good level of interaction between it and the student. They should update and maintain student models, provide examples, etc. However, it should also support a more mixed-initiative interaction. Our empirical study showed that "buggy" tutors mostly limit students to monosyllabic answers. Getting the child to interact more with the tutor is also likely to reduce the possibility of boredom.

(14) *Provide environments in which the interaction between them and students should be as close as possible to the reality of the day.* In this era, where traditional classroom instruction is still the norm and pencil-and-paper still remains the dominant technology, ITSs should reflect these realities if they are to be truly useful. It has been observed out that children experience the 'training wheels' problem when they leave, sophisticated environments and have to sit, say, our present examinations using pen and paper[27]. Nevertheless, one must not become so conservative as not to realize that the computer revolution will continue to have fundamental repercussions on the day's educational practices!

(15) *Should be easy to understand and the approach on which it is based should be transparent so that it could be applied in other suitable mathematical domains.*

DEMONSTRATION OF APPROACH

FITS, a Fraction Intelligent Tutoring System, is an example system which has been developed and which is underpinned by the above principles. Its details have been extensively discussed in other publications including[52–55]. Only an appraisal of the system against the above principles is provided here. It must be emphasized that this section is not meant to constitute the evaluation of the system. It is only meant to provide the reader with a 'feel' of how the system improves on the shortcomings pointed out. FITS has been appraised against other criteria and it is not without its limitations; a much more comprehensive evaluation of the system is provided in[54].

Example FITS protocol extract

A brief annotated and slightly modified example excerpt from an interaction between the tutor and an actual student, Rose, is illustrated by a protocol of Fig. A1, like O'Shea does with his classic Quadratic tutor[31]. Comments are in italics. The protocol is also slightly stylized, e.g. the fraction $^1/_4$ really appears in FITS as 1/4.

Discussion against principles

In this section, FITS is appraised against its principles to determine how well it achieves its objectives. Each principle is alternately examined by providing an answer to a question that concerns it. Where applicable, the reactions of students who have used FITS are also mentioned. Comparisons are also made with other tutors in the domain so as to further provide a better appreciation of the tutor.

(1) Does the system pre-model the student?
Fits does pre-model its students. It does this when the student logs on to the system for the first time. At log in, the following exchange may occur:

```
Tutor: Good afternoon. Welcome to FITS.
       Hello. What is your name?
Student:     rose.
Tutor: Fine Rose. I want to find out how much you know about
       addition of fractions. I need to know this so that I  can
       decide what and how to teach you. I can do this by having a
       look at your answers to my test of ten questions...
       Please enter the name of your file of solutions.
Student:     rose_file.
       [Pre-modelling occurs here].
...
```

FITS appears to provide a reflective enough pre-model to provide an adequate start for individualized tuition which ITSs strive to achieve, judging from its evaluation with real students. The weaker students were pre-modelled as such and instruction always began with pre-teaching some skills. On the other hand, the better ones were not pre-taught any skills. This contrasts with mal-rule ITSs, e.g. BUGGY, and with current model-tracing tutors, e.g. the GEOMETRY tutor, which do no pre-modelling at all.
(2) Does the system preteach prerequisite skills?
After pre-modelling, FITS tutors all skills or concepts/tasks that are believed to be lacking from the student's repertoire. Such tutoring may also occur after a session where the student consistently exhibits lack of certain skills. Such tutoring is mainly by exposition, examples and testing. For example, the task of adding whole numbers is taught/retaught after a session (session 2) where Rose has exhibited difficulty with it as follows:

```
Tutor: You appear to be having some difficulty with this session. I
       have noticed a skill you still have not mastered yet. Let us
       revise it again.
       Please press the <RETURN> key to continue.
Student:     <RETURN>.
Tutor: SUM WHOLE NUMBERS
       Here you just sum and record the loose or whole numbers in the
       given problem.
       EXAMPLE
       Sum the loose numbers in the problem 1:1/4 + 2:1/4.
       SOLUTION
       The sum of the whole numbers is 1 + 2 = 3.
       Another example?
Student:     no.
       [Testing of the concept occurs here].
Tutor: Let us proceed to redo session 2.
...
```

Again, current ITSs of both predominant approaches do no such pre-teaching on the basis of pre-modelling since they do not perform the latter. They therefore proceed to start tutoring different students of different aptitudes similarly. FITS improves on this limitation. In fact, it was observed that a couple of the students who used FITS benefited from this capability: they even remarked that they liked the way it taught them some of the skills that they did not know or had forgotten.
(3) Does the system provide examples (sequence of examples/lessons) such that Vanlehn's felicity conditions are satisfied?
FITS provides a sequence of examples (lessons) which attempts to satisfy [43's] felicity conditions so as to clearly communicate where and when the various skills are used. The following excerpt demonstrates how.

```
Tutor: I have arranged the examples such that you can understand how
       and when the key skills you have learnt are used. Each new
       example introduces how and when ONE of these skills is
       used. Also note how harder problems use more of the skills you
       know.
       Please press the <RETURN> key to continue.
Student:     <RETURN>.
Tutor: EXAMPLE
       Study the step by step solution to the problem 1 + 1/2.
       SOLUTION
       Note how and when you use the skill: REWRITE FRACTION.
       Solving problem ->       1 + 1/2
       Rewriting  fraction-> 1:1/2
       Please press the <RETURN> key to continue.
Student:     <RETURN>.
Tutor: EXAMPLE
       Study the step by step solution to the problem 1 + 1:1/2.
       SOLUTION
       Note how and when you use the skill: ADD WHOLE NUMBERS. You
       have already seen how and when to use the other skill.
       Solving problem ->             1 + 1:1/2
       Adding  whole  numbers  ->     2 + 1/2
       Rewriting fraction ->          2:1/2
       Please press the <RETURN> key to continue.
Tutor: EXAMPLE
       Study the step by step solution to the problem 1:1/4 + 2:1/4.
       SOLUTION
       Note how and when you use the skill: ADD EQUIVALENT FRACTIONS.
       You have already seen how and when to use the other skills.
       Solving problem ->                    1:1/4 + 2:1/4
       Adding whole numbers ->               3 + 1/4 + 1/4
       Adding  equivalent  fractions  ->     3 + 3/4
       Rewriting fraction ->                 3:3/4
       Please press the <RETURN> key to continue.
...
```

This exposition sequence could be replayed if needed and appropriate testing would ensue; obviously the pseudo-randomly generated examples will be different.

With this feature, FITS contrasts with mal-rule and model-tracing ITSs which both acknowledge the fact that students make errors in *how* perform computations but do not also seem to accede the fact that children also make arithmetical errors because of lack of knowledge of *when* to use their acquired skills/concepts. FITS improves on this shortcoming. All the students who have used FITS remarked that they liked this Vanlehn tutoring session, which seemed to clearly demonstrate to them when and where to use the various fraction skills.

(4) Does the system monitor the student step by step?

FITS definitely achieves this. The extract excerpt protocol of FITS in use depicted by Fig. A1 well demonstrates this feature which is clearly not achieved by mal-rule ITSs. Naturally, current model-tracing ITSs (e.g. the GEOMETRY tutor) support this, but a limitation has always been the fact that they could frustrate the good students who do not wish to go religiously through these step by step solutions. This shortcoming is improved on in FITS as is demonstrated in the protocol of Fig. A1. All the students really did enjoy this monitoring capability.

(5) Does the system diagnose in a problem solving context?

The reader is again referred to the exchange of Fig. A1. It clearly reveals how FITS diagnoses in a problem solving context. For example, note from the protocol of Fig. A1 how FITS diagnoses

Rose's error as having multiplied the fractions as opposed to adding them. Once more, all the students liked the diagnoses: in fact, a couple were sometimes amazed at the accuracy of FITS's diagnoses. Admittedly, mal-rule ITSs like PIXIE[50] do achieve this but probably not in such a problem solving and meaningful context as model-tracing tutors (e.g. FITS) do.

(6) Does the system enable the student to communicate his/her intention (plan) prior to executing them?

FITS certainly supports this. Figure A1 reveals that FITS regularly comes up with the question 'What operation do you want to perform?', followed by a menu of operations. The answer to this question exposes the many situations where students attempt to perform a wrong operation at a particular stage (i.e. it reveals those concepts which the student knows *how* to perform but does not understand *when* to perform them). It also makes diagnosis/remediation easier. However, as briefly mentioned previously, FITS is also flexible enough to allow the good students to directly provide any intermediate step or the final answer as demonstrated in the following exchange.

```
Tutor:  Solve 2:3/4 + 2:1/2.
        What operation do you want to perform?
        List of Operations
              1. Add equivalent fractions
              2. Cancel fraction
              3. Find equivalent fractions
              4. Sum whole numbers
              5. Change improper fraction to a mixed numbers
              6. Change mixed number to improper fraction
              7. Rewrite fraction
              8. Provide some intermediate/final answer
        WHICH <1..8>?
Student:      8.
Tutor:  Go on then.
Student:      21/4.
Tutor:  Correct. What operation do you want to perform?
        ...
Student:      8.
Tutor:  Go on then.
Student:      5:1/4.
Tutor:  Congratulations ...
```

The students enjoyed this flexibility and remarked that FITS would have been very boring without it. Naturally, mal-rule ITSs do not support such a feature. However, model-tracing ITSs do, but do not allow for the sort of flexibility demonstrated above where any intermediate/final steps could be directly given.

(7) Does the system remediate in a problem solving context?

The reader is once more referred to the exchange of Fig. A1 which clearly reveals FITS's remedial capabilities. It takes many forms, e.g. diagnosing the error, instructing the student on what to do, etc. It is worth noting the socratic remedial strategy starting and ending at the statements 'Well I think you can do with some help' and 'We return to where you were' respectively. This fragment clearly further demonstrates FITS's remedial capabilities. Such a remedial strategy is not only naturally non-existent in DEBUGGY[6] and other mal-rule ITSs, but also in model-tracing ITSs like the GEOMETRY tutor which mainly, successfully or unsuccessfully, attempt a diagnosis on any of the student's steps which do not match that of their ideal student models. The students also enjoyed FITS's remedial capabilities as it always came to their rescue when they were stuck, confused, etc. The socratic remedial strategy was a favourite with the weaker students.

(8) Does the system model the user?

FITS would hardly deserve the prefix 'intelligent' if it did not do student modelling. In the example protocol provided to supplement the answer to Question (2) above, the model revealed a 'belief' in the lack of knowledge of the task, sum whole numbers, from the student's repertoire (model) after the session. Tutoring of the task ensued; this is just one of many demonstrations that FITS does model. This contrasts with mal-rule and model-tracing ITSs which perform no modelling at all (as exemplified in Anderson's tutors) as both approaches support neither student models nor student history databases. Hence, FITS would naturally provide more individualized instruction than most mal-rule and model-tracing ITSs.

(9) Does the system test the student's understanding?

FITS's tutoring could be globally viewed as testing for all sorts of goals, e.g. testing for knowledge of *how* to use a concept, testing for knowledge of *when* to use a concept, testing to determine whether the student can handle problems of a particular difficulty level (i.e. session), etc. Hence, the exchange of Fig. A1 is the test of whether Rose can handle problems of that particular level of difficulty. A further example of testing ensues after the tutoring of the task, sum whole numbers, mentioned above [see Question (1)].

```
Tutor:  Let us see if you can do this one.
        Sum the whole numbers in the problem 3:3/4 + 4:1/6.
        Please have a guess if you are not sure.
Student:      7.
Tutor   Eh! That is good. Try this.
        Sum the whole numbers in the problem 6:4/7 + 3:5/9.
Student:      9.
Tutor:  Brilliant. I think you now understand how to perform this
        skill.
...
```

The system may also test at a later stage (after a Vanlehn tutoring session) whether the student understands when and where to use the above skill as follows:

```
Tutor:  Fine. Now show me that you understand WHEN to apply all these
        skills you have acquired.
        Assume you are at the stage 2 + 3/4 + 3.
        What operation would you perform next?
        List of Operations
              1. Add equivalent fractions
              2. Cancel fraction
              3. Find equivalent fractions
              4. Sum whole numbers
              5. Change improper fraction to a mixed numbers
              6. Change mixed number to improper fraction
              7. Rewrite fraction
        WHICH <1..7>?
Student:      4.
Tutor: Good...
```

BUGGY and other 'buggy' as well as current model-tracing tutors do carry out testing, but mainly on the overall skill: clearly, they cannot test for a subskill possibly lacking from the student's repertoire or about when to use a particular skill (as demonstrated in the previous two protocols) since they do not have an explicitly represented syllabus as in FITS, and also since they also do not possess some of its features such as modelling, pre-teaching, etc.

(10) Does the system support the various idiosyncratic ways which the student might choose to solve the problem?

FITS certainly does achieve this. For example, any of the following solutions is acceptable to FITS for the problem 2: $^1/_2$ + 3: $^1/_4$.

```
Solving problem ->                   2:1/2 + 3:1/4
Adding whole numbers ->              5 + 1/2 + 1/4
Finding equivalent fractions->       5 + 2/4 + 1/4
Adding equivalent fractions ->       5 + 3/4
Rewriting fraction ->                5:3/4
                  OR
Solving problem ->                   2:1/2 + 3:1/4
Finding equivalent fractions->       2:4/8 + 3:2/8
Adding equivalent fractions ->       2 + 3 + 6/8
Adding whole numbers ->              5 + 6/8
Rewriting fraction ->                5:6/8
Cancel fraction ->                   5:3/4
                  OR
Solving problem ->                                       2:1/2 + 3:1/4
Change mixed numbers to improper fractions->             5/2 + 13/4
Finding equivalent fractions ->                          10/4 + 13/4
Adding equivalent fractions ->                           23/4
Change improper fraction to a mixed number->             5:3/4
```

These are certainly not the only solutions to this problem that FITS would accept. Clearly, there are a multitude of other correct solutions; in fact, the students who have used FITS were observed using some of the above as well as other solutions and were pleased with this flexibility. This contrasts with the single 'optimal' solutions which Andersonian model-tracing tutors support. Mal-rule ITSs do not support such a feature. FITS also has the capability of comparing the student's solutions to that of the Problem Solving Monitor module's, with a comment on its efficiency. It can also provide a solution to any problem the student may present to it. For example, the following protocol demonstrates that FITS solves 'hard' problems which are not easily solved by human experts (this was a favourite activity amongst the students who have used FITS).

```
Solving problem ->                   67:567/765 + 75:987/789
Adding whole numbers ->              142 + 567/765 + 987/789
Finding equivalent fractions->       142 + 149121/201195 + 251685/201195
Adding equivalent fractions ->       142 + 400886/201195
Cancelling fraction ->               142 + 44534/22355
Changing to mixed number ->          142 + 1 + 22179/22355
Adding whole numbers ->              143 + 22179/22355
Rewriting fraction ->                143:22179/22355
```

(11) Does the system explicitly represent knowledge?

FITS explicitly represents its knowledge, e.g. its tutoring strategies' knowledge is largely independent of the domain-dependent knowledge to be communicated to the student. Hence, the domain-independent modules could largely be reused for some other domain; this is a similar concept to that of 'shells' in expert systems parlance. BUGGY and most mal-rule ITSs do not have any domain-independent modules (i.e. the possible domain-independent knowledge is 'hard-wired' within the rest of the code in such systems). Admittedly, some of such tutors have begun incorporating such explicit representational schemes, e.g. PIXIE[56]. Current model-tracing tutors also suffer from this shortcoming: in fact, Anderson has acknowledged this fact, and his group is currently devising a more general architecture for their model-tracing tutors. FITS definitely seems to improve on mal-rule and model-tracing ITSs as regards explicit representation of knowledge.

(12) Does the system motivate and support a more flexible style of tutoring?

FITS's style of tutoring is much more flexible because of all the various enhancements on the model-tracing paradigm, e.g. it supports various idiosyncratic solutions, it accepts various idiosyncratic inputs, it supports various tutorial strategies, etc. For the good students, FITS was observed to play more of a mentor role while for the weaker students it played a more collaborative/helpful role: this is akin to Barzilay's[33] SPIRIT system. To this degree, it is flexible. However, it falls short on motivation. The hope in FITS is to get the student to interact more in order to reduce the possibility of boredom. Such factors as boredom and motivation are probably best addressed by providing a much more intelligent interface. However, as demonstrated by the protocols observed so far, FITS clearly improves on mal-rule ITSs which tend to be very monotonous and uninteresting as their style of tutoring never changes. It also appears to improve on model-tracing tutors as they seem to be very authoritative and dogmatic in style [26], and their tutoring is also inflexible. Nevertheless, some model-tracing tutors (e.g. the GEOMETRY tutor) provide much more intelligent interfaces as this is a main tenet in Anderson's monitoring approach. In summary, FITS does appear to have improved on mal-rule and model-tracing ITSs as regards flexibility in its style of tutoring but falls well short of some of the latter as regards motivation mainly due to its fairly 'primitive' interface by today's standards. However, as will be later discussed, the students who have used it found FITS motivating enough.

(13) Does the system maintain control over the whole tutoring endeavour?

FITS certainly does maintain control; this is demonstrated by all the exchanges observed so far (e.g. Fig. A1). Too much control leads to authoritarian ITSs. However, FITS is less authoritarian in approach than current model-tracing ITSs mainly because it does not employ a tutoring strategy which constrains the student to follow some 'ideal' model.

(14) Does the system provide an environment in which the interaction between it and the student is close to the reality of the day?

FITS's monitoring capability allows for the students solving problems largely as they will with pencil and paper than current tutors do. For example, they type in the intermediate steps until they arrive at the answer, they make careless syntactic as well as logical errors just as they would with pencil and paper. This contrasts with only the final answers presented to mal-rule ITSs and the much help provided by model-tracing tutors like ALGEBRALAND [24]. In other words, the 'training wheels' problem [27] (i.e. how easily does a student cope with the problems when he/she leaves the system and returns to pencil and paper in a classroom) would probably be less for a system like FITS. In the event, a couple of the students who have used the system were, afterwards, observed solving fraction addition problems in the same manner as they were with FITS; hence, it appears FITS has largely achieved this feature.

CONCLUSION

This paper provides a rigorous critique of existing approaches to constructing intelligent tutors in mathematics. Drawing from the latter, it proposes, an alternative approach; it also provides a flavour of how this approach has been implemented in the FITS system.

Acknowledgements—The author sincerely acknowledges the invaluable comments of this supervisor, Dr Peter Coxhead of Aston University (Department of Computer Science). This research was supported by a Cameroonian Government BS grant (Cameroon Embassy, London).

REFERENCES

1. Borasi R., On the educational role of errors in mathematics: beyond diagnosis and remediation. Unpublished doctoral dissertation, State University of New York at Buffalo (1986).
2. Vanlehn K., Student modelling. In *Foundations of Intelligent Tutoring Systems* (Edited by Polson M. C. and Richardson J. J.), pp. 55–78. Lawrence Erlbaum, London (1988).
3. Burns H. L. and Capps C. G., Foundations of intelligent tutoring systems: an introduction. In *Foundations of Intelligent Tutoring Systems* (Edited by Polson M. C. and Richardson J. J.), pp. 1–19. Lawrence Erlbaum, N.J. (1988).
4. Sleeman D. H., Intelligent Tutoring Systems: a review. *Proc. EdCompCon '83 Meeting*, IEEE Computer Society, pp. 95–101 (1983).
5. Brown J. S. and Burton R. R., Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Sci.* **2,** 155–192 (1978).

6. Burton R., Diagnosing bugs in a simple procedural skill. In *Intelligent Tutoring Systems* (Edited by Sleeman D. H. and Brown J. S.), pp. 157–183. Academic Press, London (1982).
7. Attisha M. G. and Yazdani M., A micro-computer based tutor for teaching arithmetic skills. *Instruct. Sci.* **12,** 333–342 (1983).
8. Attisha M. G. and Yazdani M., An expert system for diagnosing children's multiplication errors. *Instruct. Sci.* **13,** 79–92 (1984).
9. Jones M. A. and Tuggle F. D., Inducing explanations for errors in computer-assisted instruction. *Int. J. Man–Machine Stud.* **11,** 301–324 (1979).
10. Sleeman D. H. and Smith M. J., Modelling students' problem solving. *Artif. Intell.* **16,** 171–188 (1981).
11. Sleeman D. H., Some challenges for intelligent tutoring systems. *Proc. 10th International Joint Conference on Artificial Intelligence*, Milan, Italy, Vol. 2, pp. 1166–1168 (1987).
12. Burton R. and Brown J. S., An investigation of computer coaching for informal learning activities. In *Intelligent Tutoring Systems* (Edited by Sleeman D. H. and Brown J. S.), pp. 79–98. Academic Press, London (1982).
13. Ohlsson S., Some principles of intelligent tutoring. *Instruct. Sci.* **14,** 293–326 (1987).
14. Papert S., *Mindstorms: Children, Computers and Powerful Ideas.* Basic Books, New York (1980).
15. Ridgway J., Of course ICAI is impossible . . . worse though, it might be seditious. In *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction* (Edited by Self J. A.), pp. 28–48. Chapman & Hall, London (1988).
16. O'Shea T., Evertsz R., Hennessy S., Floyd A., Fox M. and Elsom-Cook M., Design choices for an intelligent arithmetic tutor. In *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction* (Edited by Self J. A.), pp. 257–275, Chapman & Hall, London (1988).
17. Vanlehn K., Bugs are not enough: empirical studies of bugs, impasses and repairs in procedural skills. *J. Math. Behav.* **3,** 3–72 (1982).
18. Anderson J. R., Cognitive psychology and intelligent tutoring. *Proc. 6th Annual Conference of the Cognitive Science Society*, Boulder, Colo., pp. 37–43 (1984).
19. Anderson J. R., Production systems, learning and tutoring. In *Production System Models of learning and Development* (Edited by Klahr D., Langley P. and Neches R.), pp. 437–458. MIT Press, London (1987).
20. Anderson J. R. and Skwarecki E., The automated tutoring of introductory computer programming. *Commun. ACM* **29,** 842–849 (1986).
21. Anderson J. R., Boyle D. F. and Reiser B. J., Intelligent tutoring systems. *Science* **228,** 456–462 (1985).
22. Anderson J. R., Boyle D. F. and Yost G., The geometry tutor. *Proc. 9th International Joint Conference on Artificial Intelligence*, Los Angeles, Calif., pp. 1–7 (1985).
23. Kimball R. A., A self-improving tutor for symbolic integration. In *Intelligent Tutoring Systems* (Edited by Sleeman D. H. and Brown J. S.), pp. 283–307. Academic Press, London (1982).
24. Brown J. S., Process versus product: a perspective on tools for communal and informal electronic learning. *J. Educl Computing Res.* **1,** 179–201 (1985).
25. Visetti Y. M. and Dague P., Plan inference and student modelling in ICAI. *Proc. AAAI-87 6th National Conference on Artificial Intelligence*, Seattle, Wash., Vol. 1, pp. 77–81 (1987).
26. Yazdani M., Intelligent tutoring systems survey. *Artif. Intell. Rev.* **1,** 43–52 (1986).
27. Lewis M. W., Milson R. and Anderson J. R., The TEACHER APPRENTICE: designing an intelligent authoring system for high school mathematics. In *Artificial Intelligence and Instruction: Instruction and Methods* (Edited by Keersley G.), pp. 269–301. Addison–Wesley, New York (1987).
28. Self J. A., Bypassing the intractable problem of student modelling. *Proc. 1st International Conference on Intelligent Tutoring Systems*, Montréal, Canada, pp. 18–24 (1988).
29. Chan T. W. and Baskin B. A., Studying with the Prince—the computer as a learning companion. *Proc. 1st International Conference on Intelligent Tutoring Systems*, Montréal, Canada, pp. 194–200 (1988).
30. O'Shea T. and Sleeman D. H., A design for an adaptive self-improving teaching system. In *Advances in Cybernetics* (Edited by Rose J.), Gordon & Breach, London (1973).
31. O'Shea T., A self-improving quadratic tutor. In *Intelligent Tutoring Systems* (Edited by Sleeman D. H. and Brown J. S.), pp. 283–307. Academic Press, London (1982).
32. Ross P., Intelligent Tutoring Systems. *J. Comput. Assisted Learning* **3,** 194–203 (1987).
33. Barzilay A., SPIRIT: a flexible tutoring style in an Intelligent Tutoring System. In *Artificial Intelligence Applications: The Engineering of Knowledge-Based Systems* (Edited by Weisbin R. C.). IEE Computer Society (1985).
34. Wenger E., *Artificial Intelligence and Tutoring Systems.* Morgan Kaufmann, Los Altos, Calif. (1987).
35. Nwana H. S. and Coxhead P., Towards an Intelligent Tutoring System for fractions. *Proc. 1st International Conference on Intelligent Tutoring Systems*, Montréal, Canada, pp. 403–408 (1988).
36. Nwana H. S. and Coxhead P., Towards an Intelligent Tutoring System for a 'complex' mathematical domain. *Expert Syst.* **5**(4) (1988).
37. Nwana H. S. and Coxhead P., Fraction bugs: explanations, bug theories and implications on intelligent tutoring systems. *Cognitive Syst.* **2,** 275–289 (1989).
38. Nwana H. S., Intelligent Tutoring Systems: an overview. *Artif. Intell. Rev.* **4**(4) (1990).
39. Woolf B., Murray T., Suthers D. and Schultz K., Knowledge primitives for tutoring systems. *Proc. 1st International Conference on Intelligent Tutoring Systems*, Montréal, Canada, pp. 491–498 (1988).
40. Self J. A., Realism in student modelling. *Alvey-IKBS Research Workshop Tutoring Systems*, University of Exeter (17–18 November 1987).
41. Oliver W. P., Computer-assisted mathematics instruction for community college students. *Int. J. Man–Machine Stud.* **5,** 385–395 (1973).
42. Moore J. and Sleeman D. H., Enhancing PIXIE's tutoring capabilities. Technical Report AUCS/TR8709, Dept of Computing Science, Univ. of Aberdeen (1987).
43. Vanlehn K., Learning one subprocedure per lesson. *Artif. Intell.* **31,** 1–40 (1987).
44. Farell R., Anderson J. R. and Reiser B. J., An interactive computer-based tutor for LISP. *Proc. and National Conference on Artificial Intelligence-AAAI 84*, Austin, Tx, pp. 106–109.
45. Woolf B., Representing complex knowledge in an intelligent machine tutor. In *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction* (Edited by Self J. A.), pp. 3–27. Chapman & Hall, London (1988).

46. Lepper M. R. and Chabay R. W., Intrinsic motivation and instruction: conflicting views on the role of motivational processes in computer-based education. *Educl Psychol.* **20,** 217–230 (1985).
47. Self J. A., Student models in computer-aided instruction. *Int. J. Man–Machine Stud.* **6,** 261–276 (1974).
48. Sleeman D. H., UMFE: a user modelling front-end subsystem. *Int. J. Man–Machine Stud.* **23,** 71–88 (1985).
49. Woolf B., Theoretical frontiers in building a machine tutor. In *Artificial Intelligence and Instruction: Instruction and Methods* (Edited by Kearsley G.), pp. 229–267. Addison–Wesley, Reading, Mass. (1987).
50. Duchastel P., Intelligent computer assisted instruction systems: the nature of learner control. *J. Educl Computing Res.* **2,** 379–393 (1986).
51. Suppes P., The future of intelligent tutoring systems: problems and potential. *Proc. 1st International Conference on Intelligent Tutoring Systems*, Montréal, Canada (1988).
52. Nwana H. S., The anatomy of FITS: a mathematics tutor. *Intell. Tutoring Media* **1,** 82–95 (1990).
53. Nwana H. S., User modelling and user adapted in an intelligent tutoring system. *User Modell. User-Adapted Interaction* **1,** 1–33 (1991).
54. Nwana H. S., The evaluation of an intelligent tutoring system. *Intell. Tutoring Media* **1,** 117–132 (1991).
55. Nwana H. S., FITS: a Fraction Intelligent Tutoring System. *Proc. 9th National Conference on Artificial Intelligence (AAAI-91),* Anaheim, California, Vol. 1, pp. 49–54 (1991).
56. Sleeman D. H., PIXIE: a shell for developing intelligent tutoring systems. In *Artificial Intelligence and Education* (Edited by Lawler R. and Yazdani M.), Vol. 1, pp. 239–265. Ablex, N.J. (1987).

APPENDIX

[The system would begin with pre-testing and appropriate pre-teaching].

Tutor: Solve $1{:}{}^{1}/_{4} + 2{:}{}^{1}/_{2}$.

What operation do you want to perform?

List of Operations

1. Add equivalent fractions
2. Cancel fraction
3. Find equivalent fractions
4. Sum whole numbers
5. Change improper fraction to a mixed number
6. Change mixed numbers to improper fraction
7. Rewrite fraction
8. Provide some intermediate/final answer

WHICH <1..8>? *[Demanding the student's intention/plan]*

Student: 2.

Tutor: Oh no! There is no fraction to cancel.

Try again. *[Hint].*

What operation do you want to perform?

...

Student: 4.

[Could equally have chosen to find equivalent fractions or to change mixed numbers to improper fractions. Therefore the system can support various idiosyncratic ways of solving the same problem, hence no one 'correct'solution].

Tutor: Fine Rose. Perform operation.

Student: $3 + {}^{1}/_{4} + {}^{1}/_{2}$.

[Could have typed in $3{:}{}^{1}/_{4} + {}^{1}/_{2}$ or ${}^{1}/_{4} + 3{:}{}^{1}/_{2}$ or ${}^{1}/_{4} + 3 + {}^{1}/_{2}$. Note these various possible idiosyncratic inputs are possible due to explicit fractions knowledge representation.]

Tutor: Correct. What operation do you want to perform?

...

Student: 3.

Tutor: Fine. Perform operation.

Student: $3 + {}^{2}/_{8} + {}^{4}/_{8}$.

[Note these idiosyncratic inputs: could also have typed in $3{:}{}^{4}/_{8} + {}^{4}/_{8}$ or $3 + {}^{2}/_{4} + {}^{2}/_{4}$ or $3{:}{}^{8}/_{16} + {}^{8}/_{16}$.]

Tutor: Correct. What operation do you want to perform?

...

Fig. A1. *Caption on p. 43.*

```
Student:     1.
Tutor: Fine Rose. Perform operation.
Student:     3 + 6/16.
Tutor: Not Correct. You seem to have added both numerators and
       denominators to provide the numerator and the denominator of
       the fraction respectively.
       [Classic fraction error].
       Remember you add equivalent fractions by just adding the
       numerators and placing this sum over the denominator. Study
       this example to help you. 2 + 1/4 + 2/4 -> 2 + 3/4. [Hint].
```

Remember your current step is $3 + {}^2/_8 + {}^4/_8$.

```
       The operation to perform is adding equivalent fractions.
Student:     3 + 8/64.
Tutor: Still not correct. You seem to have multiplied the fractions
       together instead of adding.
       WELL I THINK YOU CAN DO WITH SOME HELP.
       Add the numerators and place the sum over the denominator for
       the problem 2/8 + 4/8.
Student:     5/8.
Tutor: Not Correct. Add the numerators: 2 + 4 = ?
Student:     6.
Tutor: Correct. Place this value 6 over the common denominator 8.
Student:     6/8.
Tutor: Correct. Now add this fraction sum 6/8 to the whole number 3.
Student:     3 + 6/8.
Tutor. Correct.
       WE RETURN TO WHERE YOU WERE.
       Remember you were at the stage: 3 + 2/8 + 4/8.
       You chose to perform adding equivalent fractions.
       Now with the help I have given you, Try again.
Student:     3 + 6/8.
       [Note the use of the explicit knowledge represented in the
       system. Also note the different tutoring strategy which FITS
       adopts here. The system could therefore switch to tutoring a
       different aspect of mathematics, if necessary, before returning
       to tutoring fractions.]
Tutor: Correct. What operation do you want to perform?
       ...
Student:     4.
Tutor: There are no whole numbers to sum Rose. Try again.
Student:     1.
Tutor: There are also no equivalent fractions to add.
       Try cancelling the fraction Rose.
       [Strategy here is to tell the student what to do].
       Remember your current step is 3 + 6/8.
       [The system notes in the student model that Rose has shown
       evidence of a lack of understanding of when to cancel a
       fraction].
Student:     2.
Tutor: Fine. Perform operation.
Student:     3 + 3/4.
Tutor: Correct. What operation do you want to perform?
       ...
```

Fig. A1. *Caption opposite.*

```
Student:      7.
Tutor:  Fine. Perform operation.
Student:      3:3/4.
Tutor:  Congratulations Rose. You have arrived at the answer.
        [After this the system would display the student's full answer,
        with a comment on its efficiency, followed by its 'optimal'
        answer.]
```

Fig. A1. Extract from protocol of FITS in use.