A Critique of Kodaganallur, Weitz and Rosenthal, "A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms"

Antonija Mitrovic, *Intelligent Computer Tutoring Group, Department of Computer Science and Software Engineering,* University of Canterbury, Private Bag 4800, *Christchurch, New Zealand*
tanja.mitrovic@canterbury.ac.nz

Stellan Ohlsson, *Department of Psychology*, *University of Illinois at Chicago, 1007 West Harrison Street, Chicago, Illinois, USA.*
Stellan@uic.edu

Research on intelligent tutoring systems (ITS) has produced many systems, but only a handful of design principles. From its start in the late 1970s, researchers have recognized that design principles should ideally be derived from psychological insights into the cognitive processes underlying the acquisition of cognitive skills, but attempts to base design philosophies on psychological principles are still few and far between (Ohlsson, 1991). Two such philosophies have come to be associated with the labels *model-tracing* (MT), which is based on the ACT-R theory of human cognition (Anderson, 2005; Anderson & Lebiere, 1998), and *constraint-based modelling* (CBM), which has grown out of our own work on learning (Ohlsson, 1993; 1996a; 1996b; Ohlsson & Rees, 1991). Given two design philosophies, it is useful to conduct systematic comparisons to ascertain the importance and implications of their differences.

In a recent IJAIED article, Kodaganallur, Weitz and Rosenthal (2005), henceforth referred to as KWR, undertake a comparison between MT and CBM. They built two tutoring systems for the same domain, one based on MT and one on CBM, and they report their observations and reflections with respect to a catalogue of issues. For every issue but one they find some weakness or potential problem with the CBM approach and also some reason for believing that any corresponding or related problem with MT will not be too difficult to overcome. They conclude that CBM tutors are "less resource intensive" to build than MT tutors, but also that MT has a wider range of application ("an MTT can be built for every domain in which a CBMT can be built, but the reverse doesn't hold") and that "the remediation provided by an MTT will be superior", the more so the more complex the learner's task (p. 141). They offer these conclusions with the purpose of providing "guidance for others interested in building intelligent tutoring systems" (p. 118).

ITS researchers should beware of their guidance, because there are several problems with their paper. The most serious is that KWR have multiple misconceptions regarding CBM which led them to make suboptimal choices in the implementation of their CBM system. As a result, several of their conclusions are wrong. Their methodology has multiple flaws. They draw sweeping conclusions about the two ITS paradigms on the basis of a limited system-building effort. The domain model for their CBM tutor contains 43 constraints (KWR, p. 128) and the model for their MT tutor contains 76 rules

(KWR, p. 137). In comparison, SQL-Tutor, a constraint-based system, contains more than 700 constraints, and the published figures for the expert modules in MT tutors for topics like geometry and programming are in excess of 400 rules. Because strengths and weaknesses of system architectures tend to be exacerbated with increased system complexity, general conclusions based on KWR's two toy systems has the potential to mislead the field of ITS research.

In the following, we group the flaws in their paper into three sections. We first address KWR's misconceptions regarding the constraint representation that is the core of the CBM approach and the suboptimal implementation decisions these misconceptions caused. In the following two sections, we discuss their conclusions with respect to the range of application and remediation. We then critique how they conducted their comparison. We end with some reflections on how this sort of comparison ought to be conducted.

MISCONCEPTIONS ABOUT CONSTRAINTS

KWR's implementation of their CBM tutor is one of many possible implementations of a CBM tutor for their domain (a miniscule subset of inferential statistics; see their Figure 1, p. 124). Their particular implementation suffers from several weaknesses.

Constraints that specify answers and actions. KWR wrote constraints as if they were rules: *When the problem is such-and-such, then the answer should be such-and-such* or *then such-and-such an operator should be executed*. That is, their satisfaction conditions specify correct answers or actions. This is true of their examples on p. 122 and in Table 3 on p.128, and they include the correct answer in their list of general features of their constraints (p. 128, near top). Furthermore, they say that in their system "the bulk of the constraints are operator constraints" (p. 130).

Forcing a rule-like programming style onto the constraint formalism is possible but negates the unique advantages of the formalism, analogous to how the implementation of Fortran-like subroutines in a rule-based system would negate modularity and other special features of such systems. Constraints provide most student modelling power when they encode principles of the domain; see Ohlsson (1993) for examples from chemistry and arithmetic. Correct problem solutions are not primarily defined by explicit representations of correct answers within individual constraints, but by the set of constraints as a whole. Correct solutions are those that conform to all the principles of the domain, i.e. that do not violate any of the constraints. When KWR claim that "in a 'pure' CBMT all the constraints are operator constraints" (p. 130), they could not be more wrong: In a 'pure' CBM tutor, *none* of the constraints would be what they call an operator constraint. All constraints would encode general principles of the domain. In living tutors, there might be some of both, but the advantages of CBM come to the fore when the bulk of the constraints in a system encode principles of the domain.

Buggy constraints. Their decision to write constraints as if they were rules lead KWR to introduce something they call 'buggy constraints' (pp. 132-133). Their example (Fig. 9, p. 133) is a constraint that has the negation of a common but incorrect answer as the satisfaction condition. Their 'buggy' constraints have the same structure as buggy rules in a rule-based system, with the relevance conditions corresponding to the left-hand sides of rules, and the content of the satisfaction conditions corresponding to the action parts of rules. Their motivation for using constraints of this sort is that "the equivalent of buggy rules – 'buggy constraints' – are required for a CBMT to provide remediation equivalent to that of the corresponding MTT" (p. 132).

This sweeping generalization does not follow from the fact that KWR decided to use such constraints. It is contradicted by the fact that we have implemented several constraint-based tutors, none of which contain any constraints similar to KWR's 'buggy constraints', and these tutors are effective (e.g. Mitrovic & Ohlsson, 1999; Mitrovic, 2003; Suraweera & Mitrovic, 2004; Martin, 2001; Mayo & Mitrovic, 2001; Martin & Mitrovic, 2003).

It is instructive to scrutinize one of KWR's many claims in this respect in detail. On page 134, KWR discuss 'buggy constraints' in the context of SQL. They consider a situation in which a student is specifying tables to be used in a query without a join condition. The authors claim that having a constraint to catch this particular situation is equivalent to having a 'buggy constraint', but this is not the case. As a proof, we present constraint 476 from SQL-Tutor below, which covers this situation. This constraint is relevant if and only if all four tests from the relevance condition are met. The first test makes sure that the FROM clause in the student's solution (SS) contains more than one table; if there is only one table, join conditions are not necessary. The second test makes sure that the FROM clause does not contain the JOIN keyword (the complementary situation is covered by another constraint). Additionally, the solutions for which this constraint is relevant should not contain a nested SELECT statement in the WHERE clause (the third test), and the ideal solution (IS) should either contain JOIN in the FROM clause or join conditions in the WHERE clause, to make sure that join conditions are necessary (the fourth test). The satisfaction condition of this constraint does not contain a correct answer, but ascertains whether the number of join conditions specified in the WHERE clause is equal to the number of tables used decreased by one.

```
(476
  "Check whether you have specified all the necessary join conditions."
  (and (> (length (find-table-names SS 'from-clause)) 1)
       (not (member "JOIN" (from-clause SS)))
       (not (member "SELECT" (where SS)))
       (or (member "JOIN" (from-clause IS))
           (> (length (join-cond (slot-value IS 'where) '())) 0)))
  (equalp (- (length (find-table-names SS 'from-clause)) 1)
    (length (join-cond (slot-value SS 'where) '())))
  "WHERE")
```

This constraint is not a 'buggy' constraint. It describes a correct domain principle. Nevertheless, it supports remediation. If it is violated, the tutor can present the associated feedback message. One can discuss exactly which feedback the student would benefit most from in this situation, but that is an empirical question to be settled by studies of learning. The CBM system builder is free to attach any feedback message to the constraint that he or she chooses. For present purposes, the main point is that constraints, although they encode correct domain principles, support remediation of errors: to provide remediation, it is enough to ascertain constraint violations and provide students with the information they need to avoid similar violations in the future. Constraints like the 'buggy' constraint in KWR's Figure 9 (p. 133) are not needed to provide remediation in a CBM tutor.

In fact, one of the great strengths of the CBM approach is precisely that there is no need to represent students' errors explicitly. The universe of possible errors is implicitly specified once all correct domain principles have been encoded into constraints: the set of errors are all the possible ways in which one or more constraints can be violated. In contrast, MT tutors are crucially dependent

upon buggy rules, typically found through empirical studies of students' mistakes. By claiming that CBM tutors need 'buggy' constraints, KWR make the two types of systems seem equal in this respect when in fact they are not: MT requires a correct rule base plus buggy rules, but CBM only requires a correct constraint base, eliminating the need for empirical studies of students' mistakes.

Lack of path constraints. Under "Capturing Planning in CBMTs," (pp. 130-131) KWR discuss how information about procedural tasks can be captured in constraints. The issue is how to provide remediation in domains in which there are strong problem-solving methods that specify the ordering, and not merely the outcome, of problem-solving steps. This issue is of sufficient importance for KWR to claim in the very second paragraph of their paper that CBM "…can be considered to be *product*-centric in that remediation is based solely on the solution state that the student arrived at, irrespective of the steps that the student took to get there" (pp. 117-118).

Once again, KWR are in error. If a domain is governed by principles that pertain to the ordering of problem solving steps, then those principles can be cast as *path constraints*. For example, if correctness of a solution crucially rests on the learner performing operation X before operation Y, then this can be captured in a constraint that says, approximately, *if operation Y was just performed, then it ought to be the case that operation X was performed before it.* KWR acknowledge this possibility, but say that using such path constraints "violates the basic spirit of the CBM paradigm" (page 131). From this they conclude that CBM cannot be successful in domains with a strong procedural flavor.

This conclusion hardly follows from the fact that KWR decided not to use path constraints in their own system. What follows is only that *their* system cannot provide remediation about procedural knowledge. We have used path constraints to great advantage in our own systems. The NORMIT system (Mitrovic 2002, 2003, 2005; Mitrovic, Koedinger, & Martin, 2003) employs precisely this technique in the very procedural domain of data normalization. In a CBM system, each constraint is applicable to a pedagogically important set of problem states. If the instructional task is procedural, then it is important to teach the student about the sequence of steps, so a CBM tutor for such a domain should contain path constraints. This does not contradict the approach, it only widens the concept of a constraint to include path constraints as well as state constraints.

KWR's advance no motivation for their resistance to the use of path constraints. They state no disadvantage or cost associated with the use of path constraints, nor do they document or even hypothesize any negative consequences of using such constraints. By refusing to use path constraints, KWR make it seem as if MT applies to procedural domains while CBM does not, while in fact both types of system apply to such domains.

KWR might have been mislead by the original formulation of the CBM approach (Ohlsson, 1992), in which one of us wondered, at a time when no CBM system had yet been build, whether CBM could work in procedural domains. However, research advances and the publication of the NORMIT system laid this concern to rest by proving that the path constraint technique works. The NORMIT system is described in Mitrovic, Koedinger and Martin (2003), Mitrovic (2002), Mitrovic (2003), and in Mitrovic et al, (2004). Furthermore, NORMIT is available on the Addison-Wesley's DatabasePlace Web portal (http://www.aw-bc.com/databaseplace/).

Explicit problem types. KWR also introduce a construct of *explicit problem type* (page 127). They conceptualise their subject matter domain as consisting of problems of distinct types, and they incorporate labels for problem types into the relevance conditions of their constraints. That is, their constraints test for, and hence only apply to, problems already assigned a pre-specified label. Their argument for this feature of their system is that "achieving this through relevance conditions would be too inefficient" (p. 127).

The first problem here is to understand the reference of "this" in the quoted sentence. What is it that cannot be accomplished via relevance conditions? What is the classification into problem types intended to accomplish? KWR do not say. The second problem is to understand what "inefficient" means in this context. The application of constraints only requires pattern matching, and contemporary pattern matchers are very fast and subject to slower-than-linear complexity functions. It is not clear what problem KWR is trying to solve by introducing explicit problem labels.

Furthermore, their claim that classification into problem types cannot be accomplished via the relevance conditions of the constraints is incorrect. We always determine *implicit* problem types through the relevance conditions, even in SQL-Tutor, which is a large system (approximately 700 constraints, as compared to 43 in KWR's system). The relevance conditions do not have to classify problems into explicitly labelled types. It is enough to specify the characteristic features of the class of problems or problem states for which the constraint is relevant.

The more important observation is that to label problems and testing for such labels in the relevance conditions is precisely what one should not do when writing constraints for a CBM tutor. Constraints are supposed to encode principles of the domain. To the extent that they do, they are *true* of all problems even if they only *apply* in a subset of problems. It is difficult to anticipate all the situations in which a domain principle might become relevant for what a student is trying to do. To limit a constraint to situations in which an explicit problem label is present cancels the flexibility that is one of the strengths of the constraint representation. In short, KWR cured their CBM system of a non-existent problem with a move that limited its usefulness.

Extraneous constraints. KWR also introduce a category of 'extraneous' constraints. Their concern in this case is that a system builder might create constraints that are unnecessary in the sense that they pertain to situations that are unlikely ever to appear. As an example, they describe a constraint that is relevant in situations in which the student has selected the *z* statistic (p. 132). To be satisfied, the constraint requires that the student has not also specified the *t* statistic. KWR cannot envision any reasoning process by which a student might come to specify both the *z* and *t* statistics at the same time, so they conclude that the constraint has no function. They then assert that when implementing a CBM system, "unnecessary effort might be expended in creating pedagogically insignificant constraints" (p. 132).

This sweeping conclusion hardly follows from the fact that KWR found such constraints in *their* system. If we take KWR at their word, their example only shows that KWR sometimes caught themselves writing bad constraints; live and learn.

However, there are reasons not to take them at their word. At the implementation level, their conclusion would only be true of their own system if KWR's interface prevents a student from specifying both statistics at the same time. With an interface that allows this, some student, somewhere, will no doubt make this error. From the pedagogical point of view, it is important for the student to know that in the relevant type of context, only one test statistic should be selected. The constraint is not 'extraneous' but captures a pedagogically significant concept, so KWR's assertion is not even true of their own example.

Even if we grant them their example, the conclusion does not follow. KWR focus on a small flaw in their own system and generalize it into a sweeping conclusion about the relative advantages of CBM and MT. But the generalization is absurd. CBM system builders with long experience of the constraint representation will not write as many extraneous constraints as newcomers to the technique like KWR. If a few such constraints slip in, they will never apply and hence will cause little trouble. On the other hand, there is no reason to believe that MT system builders do not occasionally waste

time writing unnecessary rules. KWR claim that thinking in terms of process safeguards against such mistakes, but nothing prevents the designer of a CBM system from thinking about students in terms of cognitive processes. And so on.

To summarize, KWR made several poor choices in their use of the constraint representation, most notably their use of so-called operator constraints, their introduction of so-called 'buggy' constraints, their decision not to use path constraints, and their introduction of explicit problem type labels. Each of these choices lowered the usefulness of the CBM approach or negated some advantage or strong point of CBM. The weaknesses that KWR found in their CBM system were of their own making.

RANGE OF APPLICABILITY

In their conclusion, KWR claim that "an MTT can be built for every domain in which a CBMT can be built, but the reverse doesn't hold" (p. 141). Specifically, they claim that CBM does not apply, or is less useful than MT, when (a) the goal structure for the target task is very complex, and (b) the solution to a training problem has low "information richness", i.e. little internal structure (p. 139; see also Fig. 10, p. 140).

This claim is false in both parts. First, with respect to complexity, KWR write "As the problem goal structure increases, it becomes increasingly difficult for a CBMT to fathom where in the problem-solving process the student has made an error and in turn it becomes increasingly difficult for a CBMT to offer targeted remediation" (p. 139). This statement is utterly at variance with how a CBM system works: After each student step, the relevance conditions are matched against the current state of the student's problem solving effort; for those that match, the satisfaction conditions are matched, and any violations are noted. In other words, a CBM tutor detects a student error immediately after any step that violates a constraint. It is not difficult "to fathom where in the problem-solving process the student has made an error" because the student's behaviour is monitored continuously and errors are caught as they are made. Constraints are matched against the state of the problem solving effort after every step, so CBM is not affected by the complexity of the goal structure. We note that this is also true of an MT system that matches production rules to problem states and students' overt[1] actions.

Second, KWR claim that the "constraint-based paradigm is feasible only for domains in which the solution itself is rich in information. There is no such restriction for model-tracing" (page 118). To support this assertion, KWR consider statistics problems with answers like "reject/do not reject" the null hypothesis (p. 131) and physics problems that might have an answer like "70 mph" (p.139). Their claim is that CBM cannot operate in such domains because there is not enough internal structure in the answer on which to base a tailored feedback message. Contrary to this claim, existing CBM tutors work just fine in domains with information-impoverished answers. For example, CAPIT (Mayo & Mitrovic, 2001) is a constraint-based tutor which teaches punctuation and capitalization rules in English. The tasks CAPIT gives to students consists of one or more sentences that are to be revised, and the result of the appropriate revisions do not have any inherent structure; for example, a capital letter does not have any meaningful internal structure. Another example is LBITS (Martin & Mitrovic,

[1] If an MT system attempts to infer the unobserved mental steps that occurred between one overt step and the next, then that system is subject to a combinatorial explosion with a magnitude that is directly proportional to the length of the sequence of inferred steps. We understand that some MT implementation efforts are moving in this direction (V. Aleven, personal communication). If so, the resulting systems will be adversely affected by the complexity of the domain.

2003), a constraint-based tutor that teaches vocabulary skills in English. Example tasks from LBITS include turning singular forms of nouns into plural, comparing adjectives and generating nouns from verbs. In both of these tutors, the solutions have little internal structure. Nevertheless, these constraint-based tutors are effective (Martin, 2001; Mayo & Mitrovic, 2001; Martin & Mitrovic, 2003). The constraints in LBITS and CAPIT look for patterns in various groups of words, and check whether students use these patterns.

There might be situations in which there is *neither* a student behaviour *nor* any internal structure in the answer. But if the student provides no path information and an answer with no internal structure such as *Yes/No*, then neither MT nor CBM could carry out a meaningful cognitive diagnosis. Even a human tutor would find this situation opaque, and the only reasonable feedback might be *Your answer is correct/incorrect*. In any case, this scenario does not differentiate between MT and CBM.[2]

KWR's attempt to argue that MT applies across a wider range of domains than CBM is misguided. The truth is the exact opposite. There are two differences. MT only applies when the system builder can specify an expert or ideal student model that can solve the problems presented to the student. Without correct rules, an MT system cannot recognize correct steps and solutions. There is no such limitation on CBM, as we have demonstrated by developing intelligent tutors for design tasks in the area of SQL, database design and UML (Baghaei & Mitrovic, 2005; Baghaei, Mitrovic, & Irwin, 2005). Domains like these cover design tasks for which no problem-solving algorithms exist, and therefore no expert models are available.

Furthermore, MT only applies when there is some behaviour to trace. It is no accident that the MT approach has primarily been applied to domains with strongly sequential solutions such as algebra, geometry, physics and programming (KWR, p. 119). Whether the final answer is information rich or information poor, the MT approach only applies in a straightforward way when the student generates a sequence of steps. If not, to what would an MT system match its correct and buggy rules? ("…MTTs can provide well-targeted remediation only when one or more buggy rules are used in a successful trace…"; KWR, p. 121). This limitation does not apply to CBM, precisely because constraints are evaluative rather than generative in nature. When a behavioural trace is available, a CBM system can make use of it by matching path constraints against it; when it is not available, a CBM system can make do by matching state constraints against the final answer. The latter option is not open to MT. It is therefore CBM that has the wider range of application.

REMEDIATION

Although none of their formally stated conclusions pertain to remediation, concerns about remediation pervade KWR's paper. They repeatedly claim that the "quality" of the remediation (feedback) delivered by an MT system is "superior" to the remediation of the corresponding CBM system (unless the latter is deliberately designed to mimic the former; see p. 131, near bottom). They even go as far as claiming that this is "generally accepted" (p. 139).

KWR do not define what they mean by superior quality of remediation, nor do they provide an example. It would have been instructive if they had selected a student error and presented the relevant feedback delivered by their two systems side by side, to allow their readers to judge the relative quality. Although they exhibit the feedback messages associated with one of their constraints (Table 3,

[2] Bayesian knowledge tracing can make use of multiple right/wrong answers to update a Bayesian network model of a student.

p. 128), KWR do not provide a single example in their paper of a feedback message output by their MT system. One plausible interpretation of "superior" is that superior remediation support faster, better or deeper learning, but KWR do not present any data from users interacting with their systems. Instead, their treatment of this issue consists of a series of assertions.

One assertion pertains to the possibility of giving meaningful feedback when the final answer a student submits is information poor (p. 131). In such domains, KWR write, a CBM system "can evaluate its correctness, but has no information on which to base remediation" (p. 131). We have already shown above that this is a concern only because KWR do not use path constraints. The success of the NORMIT system shows that procedural remediation based on path constraints works (Mitrovic, Koedinger & Martin, 2003; Mitrovic, 2002; Mitrovic, 2003; Mitrovic et al, 2004; Mitrovic, 2005).

KWR also claim that "the equivalent of buggy rules – 'buggy constraints' – are required for a CBMT to provide remediation equivalent to that of the corresponding MTT" (p. 132). We have already discussed above why 'buggy' constraints are unnecessary. Constraints encode principles of the domain; incorrect solutions violate one or more constraints; remediation focuses on teaching the student the principle encoded in the constraint. The fact that existing CBM systems do not use any buggy constraints proves that such constraints are not needed for effective remediation.

KWR worry that a CBM system might provide "misleading remediation when the student has provided an unexpected but correct solution" (p. 135). This is not a concern with CBM. By definition, correct behaviour does not violate the principles of the domain. If the constraints are written to encode the domain principles, then the constraint base will recognize any correct behaviour as correct, no matter how unexpected, by the fact that it does not violate any of the constraints. (If the constraints are written KWR style, as if they were operator rules, this might no longer hold.) KWR fails to mention that the possibility of correct but unanticipated student solutions is a concern with MT: because an MT system relies on matches to correct rules to recognize correct task behaviour, such a system can only recognize correct solutions that the system builder has anticipated. Thus, an MT system might make the pedagogical mistake of trying to tutor a student away from a creative solution. The potential to recognize creative but correct solutions is one of the great advantages of the constraint formalism.

KWR also raise the problem of what remediation to provide when a student solution violates multiple constraints (p. 134). There are indeed cases in which an incorrect step can be a symptom of more than one underlying misconception. KWR claim that this problem does not arise with MT tutors, but this is not so. In a real MT system with hundreds of buggy rules, it is possible for several rules to match a given student error. For example, consider the problem 426 – 76 = ? and the incorrect answer "450." At the point when the student enters "5" in the second column, his or her behaviour does not differentiate between SMALLER-FROM-LARGER and BORROW-NO-DECREMENT, two standard subtraction bugs (VanLehn, 1990). The problem of what remediation to provide when there are multiple interpretations of a given incorrect step or answer does not differentiate between CBM and MT, but is equally problematic for both. Indeed, this problem applies equally to all student modelling techniques, because it is ultimately rooted in the difficulty of making fine grained inferences about cognitive structures and processes on the basis of observations of behaviour, especially as viewed through the narrow bandwidth of human-computer communication.

Let us next consider the case in which a student violates multiple constraints because he or she really has multiple misconceptions about the subject matter. This is a pedagogical, not a student modeling problem. Multiple responses to this situation have been explored in ITS research. A student can be given multiple error messages, or the pedagogical module can have an algorithm for choosing which constraint is most important to remediate. The latter can be done off-line, via domain expert

rankings of the importance and centrality of constraints, or it can be done dynamically, by calculating a measure of which constraint the student needs to practice most. This problem too applies equally to CBM and MT systems because CBM and MT are approaches to student modelling and the problem of which misconception to tutor first is a problem of pedagogical decision making. In short, KWR's treatment of the problem of multiple cognitive diagnoses (p. 134) claims that it only arises in the context of CBM, but in fact it does not differentiate between CBM and MT systems.

KWR repeatedly question whether CBM can provide "equivalent remediation," as compared to an MT system. They even say that they focus on "equivalences between rules and constraints in order to provide a finer grained comparison" (p. 130). This is not the optimal way to capture the unique strengths of these approaches. The CBM approach was not invented to provide feedback that is "equivalent" to the feedback delivered by an MT system, but to do something different. Because CBM catches errors via constraint violations, a CBM tutor in which the constraints encode domain principles can talk to students about those principles. If constraint <C, R> is violated, it is easy to tell the student, "you know, when C is the case, R needs to be the case as well." We believe that instruction that focuses on the basic concepts and principles of a domain is of superior quality, as compared to instruction that talks about this or that particular incorrect response to a particular problem.

Ultimately, the quality of different types of remediation is an empirical issue that has to be settled by measures of student learning. If students learn faster, more or better with one style of remediation than another, then that style is superior. KWR do not report any data from interactions between learners and their systems.

In short, KWR present no definition of quality remediation, no examples of such remediation, and no data to support their claim of superiority. The rhetorical function of this claim within their paper is illustrated by the following quote: " …we observe that the development effort required to build a model-tracing tutor is greater than that for building a constraint-based tutor. This increased effort is a function of additional design requirements that are responsible for the improved remediation" of MT tutors (p. 118). That is, the assertion of superior remediation allows KWR to dismiss their one positive observation about CBM, namely that the development effort for a CBM system is less than that of the corresponding MT system, by claiming that the extra effort of implementing an MT system pays off pedagogically.

METHODOLOGY

There are also flaws in KWR's comparison methodology. Because KWR presume to be guiding the ITS field, it is important to point these out to IJAIED readers. Each of these recurs repeatedly, but for the sake of brevity, we sometimes limit ourselves to a single example.

Unwarranted Generalizations. As we have documented above, KWR repeatedly proceed by describing a flaw in their own CBM system, and then claiming that it illustrates a principled weakness with the CBM approach. For example, they find what they regard as an unnecessary constraint in their system, and then claim, without any further evidence or argument, that the CBM methodology will somehow cause system builders to spend significant amounts of time creating such 'extraneous' constraints. The repeated use of unwarranted generalizations becomes particularly gregarious when the feature being generalized is of KWR's own invention. They write constraints with correct answers and actions as satisfaction conditions, then claim that this is how constraints must be. They find

themselves using 'buggy' constraints, and then claim that CBM systems require such constraints. They decide not to use path constraints, and then write as if this was a necessary feature of the CBM approach. And so on. In no instance does KWR provide any evidence that the reported flaws in their CBM system are weaknesses of the CBM approach rather than of their own implementation effort. Because KWR claim to be conducting a systematic comparison between CBM and MT it is relevant to notice that there is no single instance in their paper where KWR generalize from a flaw in their MT system to a principled weakness of the MT approach. (They do, however, sometimes generalize a good feature of their MT system into a positive claim about that approach; see p. 134.)

Incomplete comparisons. KWR's analysis takes the form of a catalogue of issues. On several issues, they only discuss one side of the supposed comparison. For example, consider the KWR treatment of the system development effort. An MT system requires a library of buggy rules to match against the students' solution path. Bug libraries can only be constructed on the basis of empirical studies of student errors, they tend to be labour intensive and they may or may not transfer from one student population to another. CBM does not require a bug library. Once the constraint base has been written, the set of all possible errors has been implicitly specified: it consists of all ways to violate one or more constraints. The need for bug libraries is a dimension of comparison on which CBM is clearly superior to MT. KWR fail to mention this rather important difference between the two types of systems.

As a second example, consider the subsection titled "A comparison of remediation provided by two tutors." KWR complain about what they see as bugs[3] in SQL-Tutor (p. 136). They say that these bugs "provides confirming evidence" regarding "the difficulties of providing targeted remediation using the CBMT paradigm" (p. 136). The existence of a few bugs is not controversial. We have pointed out in several papers that the domain model of SQL-Tutor (i.e., the set of constraints) is not yet perfect. At 700 constraints and counting, it is still incomplete and a handful of bugs are bound to remain. However, KWR's claim that a handful of bugs in SQL-Tutor confirms that there are difficulties with the CBM approach is absurd. The possibility of bugs in large knowledge bases is not an issue that differentiates CBM and MT. When an MT system is based on several hundred production rules rather than the 76 rules in KWR's system, it, too, will contain bugs and have to go through a period of debugging. If the occurrence of bugs in a particular CBM system is evidence against the CBM approach, then the occurrence of bugs in the rules of any MT system is equally strong evidence against the MT approach. Obviously, incorrect domain rules will harm the functioning of an MT system. The possibility of mistakes when building the expert domain model applies equally to every type of ITS system, but KWR write as if this is a problem with CBM only.

KWR say that their constraints can calculate parts of the solution, and that therefore the ideal solution does not have to be specified. This is not novel; we have used precisely this technique in NORMIT (Mitrovic 2002, 2003, 2005). However, this technique can only be used in domains for which well-defined problem solving procedures exist. In domains such as SQL this is not possible. CBM nevertheless works very well in that domain. KWR fail to mention that an MT tutor would be difficult to build for a task domain for which an expert or ideal student rule base cannot be identified.

---

[3] KWR are wrong to regard some of these features of SQL as bugs. For example, misspelling a table name is a true mistake in the SQL domain. No database management system is able to deal with queries with misspelled names. Therefore, students have to be told about this type of mistake. Likewise, using double quotes instead of single quotes is a true error. Students must learn these syntax rules if they are to use SQL in a real context. However, this is a quibble compared to the principled flaws that we highlight in the main text.

Use of the Literature. KWR frequently cite the ITS literature in support of their conclusions. However, they do so selectively and, in a few cases, inappropriately.

KWR do not mention NORMIT, our data normalization tutor. If NORMIT is included in the comparison, many of their arguments against CBM can be seen to be invalid. Most importantly, NORMIT operates in a procedural task domain, disproving KWR's claim that CBM cannot be applied in such domains. The authors must have known about this constraint-based tutor: it was discussed in a paper that they refer to several times (Mitrovic, Koedinger, & Martin, 2003). NORMIT is also presented in other recent papers that would turn up in any literature search on intelligent tutoring systems (Mitrovic, 2002; Mitrovic, 2003; Mitrovic et al, 2004; Mitrovic, 2005). Furthermore, NORMIT is available on the Addison-Wesley's DatabasePlace Web portal (http://www.aw-bc.com/databaseplace/), which KWR describe on pages 135-136. KWR say they used SQL-Tutor on this Web portal, so the portal was neither unknown nor inaccessible to them.

KWR do not cite our journal article about KERMIT (Suraweera & Mitrovic, 2004), although this paper was available before KWR's paper was published. The former provides a discussion of the types of domains to which constraint-based modelling is applicable as well as details of system implementation and evaluation, all topics that KWR found important enough to discuss in their paper. Furthermore, we have developed an authoring tool called WETAS. KWR refer to an early paper on WETAS, but do not mention the later and readily available papers presented at AIED 2003 and ITS 2004 (Martin & Mitrovic, 2003; Suraweera, Mitrovic & Martin, 2004a, 2004b).

KWR support their claim that an MT tutor can be build for every domain in which a CBM tutor can be build with a reference to Mitrovic, Koedinger and Martin (2003): "Indeed the comparison provided by Mitrovic et al. (2003) was predicated on building an MTT for an existing CBMT" (page 141). This is not an accurate report on the content of the cited paper. It is true that the work reported in that paper consisted of building a prototype MT tutor that covered a small part of the domain covered by our CBM database design tutor, KERMIT, but the paper does not claim or argue that it is always possible to do so. In that paper, we said that an MT tutor cannot be built for domains for which there is no known or workable problem-solving method. It is not possible to develop a model-tracing tutor for database design, but CBM systems work well in such domains. The point argued in the cited paper was therefore the opposite of the point that KWR claim that it supports.

There are other instances of such misuse of quotes. KWR repeatedly quote Mitrovic, Koedinger and Martin (2003) and (Martin, 2001). In each case, they locate some sentence in these works that contains a negative-sounding phrase and then quote that sentence in support of their position: "less comprehensive" (see quote on p. 132), "misleading remediation" (quote on p. 135), and "compromises cognitive fidelity" (quote on p. 139) are examples. In each case, the quote is inappropriate in the sense that the cited paper does not argue the point that KWR use the quote to support. Mitrovic et al (2003) do not claim that remediation by CBM is necessarily or in general less comprehensive than remediation by MT, nor that it necessarily compromises cognitive fidelity. Likewise, Martin (2001) did not argue that CBM systems have a high probability of providing misleading remediation.

On page 139, KWR say: "It is generally accepted that MTTs outperform CBMTs with respect to remediation quality." Significantly, KWR provide no references to support this assertion. There are only a few hundred ITS researchers in the world, and only a few of them have had occasion to investigate the relative advantages of CBM and MT. KWR's article is the only publication other than Mitrovic et al. (2003) to compare the two, so there is no history of public debate on this matter. Who are all these people who "generally accept" KWR's view?

CONCLUSION

KWR's flawed comparison between MT and CBM has the potential to mislead readers of IJAIED who do not have experience with either design philosophy. KWR's lack of knowledge about CBM, their limited ITS experience, their misunderstandings of the constraint formalism, their unwarranted generalizations, their selective and incomplete comparisons and their inappropriate literature citations provide no basis for understanding or evaluating the relative strengths and weaknesses of the CBM and MT approaches to ITS design.

How should a comparison between alternative design philosophies for ITSs be conducted instead? One answer is that they should be based on the intended pedagogical outcome: if learners learn more, better or faster with one type of system than another, then this supports the underlying design philosophy. However, the prospect of carrying out such comparisons with precision is dimmed by the difficulty of measuring learning outcomes: how do we separate the effects of the system design from, for example, the depth and accuracy of the underlying subject matter analysis, the superior pedagogical insight and wisdom embedded in the formulation of the feedback messages, or some other factor that is independent from design philosophy? In addition, given the possibility that people learn different subject matter representations through different styles of instruction, it is difficult to ensure that all relevant outcome dimensions are assessed: Should the outcome measures include, for example, the ability to explain why certain problem solving steps are right or wrong? What about the ability to transfer to tasks outside the primary task domain? How do we know that we have measured all relevant outcome variables? In spite of such difficulties, the ability to support learning is the ultimate ground for evaluation.

Awaiting such empirical comparisons, ease of use remains important. For system builders, it is not a trivial question whether a particular design philosophy makes them do more or less work. Because authoring tools will soon be generally available for both MT and CBM, this question boils down to the relative amount of work required to write a rule base versus a constraint base for a given subject matter domain. In this comparison, CBM is clearly superior: CBM requires only a representation of the correct knowledge of the domain, while an MT tutor requires both an expert model and a set of buggy rules. This is the one and only dimension of comparison for which KWR provide quantitative evidence: Their CBM system required 43 constraints (p. 128) and took 18 days to build (p. 137), while their MT system required 76 rules and took four months to build (p. 137). We note that these measures do not include the work required to identify the relevant student mistakes. KWR say nothing about how they decided which buggy rules their system would need. Buggy rules typically have to be identified in a time consuming empirical study, which, if it had been included in KWR's accounting of their development efforts, would have widened the lead of CBM over MT yet further. Moreover, what little evidence there is on the matter suggests that buggy rule libraries do not transfer well between different student populations. This threatens MT tutor builders with the need for additional empirical work and some retuning every time an MT system is moved into a new instructional context or is confronted with a new student population. Contrary to KWR's misconception, CBM does not require any ‘buggy constraints’ because the universe of student errors is implicitly specified as the set of all possible ways in which a student can violate one or more constraints, provided that constraints are written to encode domain principles rather than correct solutions to specific exercises. This key feature is a fantastic advantage of the constraint formalism over rules, and we expect that ITS researchers will want to exploit it fully. We invite researchers to get in touch with us if they want to discuss the implementation of their CBM systems.

REFERENCES

Anderson, J, R. (2005). Human symbol manipulation within an integrated cognitive architecture. *Cognitive Science*, 29, 313-341.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Baghaei, N., Mitrovic, A., & Irwin, W. (2005). A Constraint-Based Tutor for Learning Object-Oriented Analysis and Design using UML. In C.K. Looi, D. Jonassen & M. Ikeda (Eds.) International Conference on Computers in Education ICCE 2005 (pp. 11-18).

Baghaei, N., & Mitrovic, A. (2005). COLLECT-UML: Supporting individual and collaborative learning of UML class diagrams in a constraint-based tutor. In R. Khosla, R. J. Howlett & L. C. Jain (Eds.) Proceedings of KES 2005 (pp. 458-464). LCNS 3684. Berlin: Springer-Verlag.

Kodaganallur, V., Weitz, R., & Rosenthal, D. (2005). A Comparison of Model-Tracing and Constraint-based Intelligent Tutoring Paradigms. *International Journal of Artificial Intelligence in Education*, 15(2), 117-144.

Martin, B. (2001). *Intelligent Tutoring Systems: The Practical Implementation of Constraint-Based Modelling*. Ph.D. Thesis, University of Canterbury, New Zealand.

Martin, B., & Mitrovic, A. (2003). Domain Modeling: Art or Science? In U. Hoppe, F. Verdejo & J. Kay (Eds.) *Proceedings of the 11th International Conference on Artificial Intelligence in Education AIED 2003* (pp. 183-190). Amsterdam: IOS Press.

Mayo, M., & Mitrovic, A. (2001). Optimising ITS Behaviour with Bayesian Networks and Decision Theory. *International Journal of Artificial Intelligence in Education*, 12(2), 124-153.

Mitrovic, A. (1998). Experiences in Implementing Constraint-Based Modeling in SQL-Tutor. In B. Goettl, H. Halff, C. Redfield & V. Shute (Eds.) *Proceedings of ITS 1998* (pp. 414-423). Berlin: Springer-Verlag.

Mitrovic, A. (2002). NORMIT, a Web-enabled tutor for database normalization. In Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson & C.-H. Lee (Eds.) *Proceedings of the International Conference on Computers in Education* (pp. 1276-1280). December 3-6, 2002, Auckland, New Zealand.

Mitrovic, A. (2003). Supporting Self-Explanation in a Data Normalization Tutor. In V. Aleven, U. Hoppe, J. Kay, R. Mizoguchi, H. Pain, F. Verdejo & K. Yacef (Eds.) *Supplementary proceedings AIED 2003* (pp. 565-577).

Mitrovic, A. (2005). The Effect of Explaining on Learning: a Case Study with a Data Normalization Tutor. In C-K Looi, G. McCalla, B. Bredeweg & J. Breuker (Eds.) *Proceedings of Artificial Intelligence in Education AIED 2005* (pp. 499-506). Amsterdam: IOS Press.

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a Constraint-Based Tutor for a Database Language. *International Journal of Artificial Intelligence in Education*, 10(3-4), 238-256.

Mitrovic, A., Koedinger, K., & Martin, B. (2003). A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling. In P. Brusilovsky, A. Corbett & F. de Rosis (Eds.) *Proceedings of the 9th International Conference on User Modeling UM 2003* (pp. 313-322). LNAI 2702. Berlin: Springer-Verlag.

Mitrovic, A., Suraweera, P., Martin, B., & Weerasinghe, A. (2004). DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. *Journal of Interactive Learning Research*, 15(4), 409-432.

Ohlsson, S. (1991). System hacking meets learning theory: Reflections on the goals and standards of research in Artificial Intelligence and education. *Journal of Artificial Intelligence in Education,* 2(3), 5-18.

Ohlsson, S., & Rees, E. (1991). The function of conceptual understanding in the learning of arithmetic procedures. *Cognition & Instruction*, 8(2), 103-179.

Ohlsson, S. (1992). Constraint-based student modeling. *Journal of Artificial Intelligence and Education*, 3(4), 429-447.

Ohlsson, S. (1993). The interaction between knowledge and practice in the acquisition of cognitive skills. In A. Meyrowitz & S. Chipman (Eds.) *Foundations of knowledge acquisition: Cognitive models of complex learning* (pp. 147-208). Norwell, MA: Kluwer Academic Publishers.

Ohlsson, S. (1996a). Learning from performance errors. *Psychological Review*, 103, 241-262.

Ohlsson, S. (1996b). Learning from error and the design of task environments. *International Journal of Educational Research*, 25(5), 419-448.

Suraweera, P., & Mitrovic, A. (2004). An Intelligent Tutoring System for Entity Relationship Modelling. *International Journal of Artificial Intelligence in Education*, 14(3-4), 375-417.

Suraweera, P., Mitrovic, A., & Martin, B. (2004a). The use of ontologies in ITS domain knowledge authoring. In J. Mostow & P. Tedesco (Eds.) Proceedings of the 2nd International Workshop on Applications of Semantic Web for E-learning (pp. 41-49). ITS 2004 conference, Maceio, Brazil.

Suraweera, P., Mitrovic, A., & Martin, B. (2004b). The role of domain ontology in knowledge acquisition for ITSs. In J. Lester, R. M. Vicari & F. Paraguaçu (Eds.) *Proceedings of the 7th International Conference on Intelligent Tutoring Systems ITS 2004* (pp. 207-216). LNCS 3220. Berlin: Springer.

VanLehn, K. (1990). Mind Bugs: The Origins of Procedural misconceptions. Cambridge, MA: The MIT Press.