

# An Assessment of Constraint-Based Tutors: A Response to Mitrovic and Ohlsson's Critique of "A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms"

**Viswanathan Kodaganallur, Rob R. Weitz and David Rosenthal**, *Department of Computing and Decision Sciences, Stillman School of Business, Seton Hall University, South Orange, NJ 07079, USA*  
*weitzrob@shu.edu*

**Abstract.** Model tracing and constraint-based modeling are two prominent paradigms on which intelligent tutoring systems (ITSs) have been based. We Kodaganallur, Weitz and Rosenthal (2005), have written a paper comparing the two paradigms, and offering guidance based on this comparison to prospective authors of ITSs. In a detailed critique of our paper, Mitrovic and Ohlsson (2006) have taken issue with many of our observations and conclusions. In this work we refute their critique and provide a more general, critical assessment of constraint-based modeling.

## INTRODUCTION

Model tracing (MT) and constraint-based modeling (CBM) are two prominent paradigms on which intelligent tutoring systems (ITSs) have been based. We, Kodaganallur, Weitz and Rosenthal (2005), hereafter KWR, have written a paper comparing the two paradigms, and offering guidance based on this comparison to prospective authors of ITSs. In a detailed critique of our paper, Mitrovic and Ohlsson (2006), hereafter MO, have taken issue with many of our observations and conclusions. In this work we refute their critique and provide a more general, critical assessment of CBM.

MO point out what they consider to be numerous flaws in KWR's paper. We list below in Table 1 their most important claims and how this paper addresses them. All the issues raised by MO are addressed in detail in subsequent sections of this paper.

Table 1  
Summary of major points raised by MO and the responses presented here

<b>Claim</b>	<b>How we address it</b>
KWR performed its comparison based on very small systems.	We show that for the conclusions KWR draw, system size is unimportant. (Indeed, MO never indicate which of KWR's claims are invalid due to system size.) Further, the number of constraints in KWR's CBMT is of the same order of magnitude as most of the CBMTs reported in the literature.

## Claim

## How we address it

KWR's constraint representation is faulty and this led to many of their findings about the disadvantages of CBM. Specifically:

KWR foisted a rule-like style onto the constraint formalism

KWR wrongly suggest that CBM might need to use *buggy constraints*, the equivalent of buggy rules in MTT

KWR suggest problems with the application of CBM to procedural domains because they did not understand *path constraints* and did not make use of these.

KWR are wrong when they say that CBM is at a disadvantage in domains where the solution state is information-impoorished and/or when the underlying goal structure of the task is complex.

We show with several clear examples how KWR's constraint representation is in strict accordance with suggested CBM practice.

We also show that the one constraint that MO chose to showcase actually violates the spirit of CBM – that a constraint should encapsulate general domain principles.

We show with examples how a CBMT will be unable to distinguish between some instructionally significant student actions in the absence of such constraints.

We point out that KWR were indeed forced to use such constraints in their procedural domain application. However, we draw heavily from the CBM literature to establish that the spirit of CBM actually proscribes the use of such constraints and show that the use of such constraints merely leads to a CBMT mimicking an MTT. The very term *path constraint* is an oxymoron.

Although MO paint a picture of their having been aware of path constraints and how these can be used to apply CBM to procedural domains, we point out that the issue of procedural domains is addressed only cursorily in the CBM literature until the arrival of NORMIT. Further, NORMIT deals only with a fixed-sequence procedural problem (where dealing with procedure is trivial), and aside from partitioning the problem into this fixed-sequence, the mechanism for handling procedure is not described in the NORMIT literature.

We provide concrete examples of situations where the solution state is information impoverished and show that a CBMT cannot be as discriminating as a corresponding MTT in such situations.

We also provide evidence from the published literature where the remediation on the goal structure is shown to be very important – this is the kind of remediation that CBM finds very difficult to provide.

We show concrete examples from SQL-Tutor, the most widely reported CBMT, and demonstrate that it generally is unable to fathom where in the solution process the student is.

We then refute MO's argument that the CBM camp has demonstrated that CBM is applicable to design tasks – tasks for which no well-defined solution procedures exist and thus

**Claim****How we address it**

MT is inherently inapplicable. We show that the subset of tasks that they take from these complex domains do not represent the complexities of design tasks. Well-defined solution procedures can in fact be created for these simplified tasks, and hence MTT's can be built for them.

KWR wrongly claim that the remediation provided by CBMTs is inferior to that provided by corresponding MTTs.

We show through concrete evidence culled from prior sections and by logical argument why CBM is at a disadvantage with respect to remediation quality. In addition, we provide telling examples from SQL-Tutor.

KWR incorrectly and misleadingly quote from the literature.

We refute each claim citing the charge by MO, what KWR actually say, and what the literature referenced by KWR says. In two cases we suggest minor rewordings.

CBM is certainly an attractive idea – representing domain principles as constraints appears to be considerably easier than building a set of complex expert and buggy production rules. It apparently eliminates the need to conduct extensive studies of student errors and the need for an expert solver. Our findings indicate that there are fundamental issues that affect the universal applicability of the paradigm and that representing domain concepts as constraints presents significant challenges.

MO divide their critique of KWR into five sections a) introductory remarks, b) misconceptions about constraint representation, c) range of applicability, d) remediation, e) methodology and f) conclusion. We shall take the major points discussed by MO one by one; we begin by addressing their introductory remarks.

**System Size**

MO begin by challenging the conclusions drawn by KWR based on the size of the systems KWR built to perform the comparisons. It is important for the reader to note that MO never actually specify which of KWR's claims are invalidated by the size argument. On this basis alone, the charge should be dismissed, but it is illuminating to pursue this further.

MO state (p. 278):

"Because strengths and weaknesses of system architectures tend to be exacerbated with increased system complexity, general conclusions based on KWR's two toy systems have the potential to mislead the field of ITS research..."

This claim may seem damaging, but is, quite clearly, irrelevant because KWR do not compare the strengths and weaknesses of system *architectures*. For example, KWR do not make any observations about the efficiency of the two systems, or about the relative difficulties of maintaining the knowledge bases – two among many aspects that could indeed be affected by the size of a system. And, in fact, we are not aware of any architecture problems that exist in small systems and then magically disappear with increased system size.

Beyond this, the observations in KWR stem from conceptual and logical arguments about the knowledge representation mechanisms used by the two paradigms, and are drawn from experience gained from building the two tutors. KWR provide abundant supporting evidence in each case.

MO correctly state that (p. 277):

"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."

However, MO omit the following from consideration. Mitrovic's own comparison of the two paradigms (Mitrovic, Koedinger, & Martin, 2003) contrasted a model-tracing tutor (MTT) composed of 25 production rules that correspond with just 23 constraints of an already existing constraint-based tutor (CBMT). The full-blown CBMT in question here (KERMIT) had some 90 constraints – not much bigger than the CBMT in KWR. CAPIT, the CBMT for capitalization and punctuation, has 25 constraints (Mayo & Mitrovic, 2001; Mayo, Mitrovic, & McKenzie, 2000). NORMIT has 53 constraints (Mitrovic, 2002). So, the hypothesis testing CBMT described in KWR is comparable in size to tutors Mitrovic and others in the CBM camp have written papers about and drawn conclusions from for years.

Therefore it seems that either the system-size bar required to draw conclusions is movable, or that MO's argument is ineffectual.

## **Constraint Representation**

MO claim weaknesses in KWR's representation of constraints under five sub-categories. We consider each in turn.

Constraints that specify answers and actions: MO state (p. 278),

"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)."

In fact, KWR write constraints exactly as CBM experts have reported.

Below is an example constraint from Mitrovic et al. (2003):

Cr: A base angle of an isosceles triangle is known ( $\theta_1$ )

And the student has calculated the size of the other base angle ( $\theta_2$ )

Cs: The size of  $\theta_2$  is  $\theta_1$

The above constraint also encodes the correct answer (that the correct value of  $\theta_2$  is  $\theta_1$ ) in the satisfaction condition. This is true of other constraint examples from the published literature. (The reader may simply refer to Appendix A which contains the complete triangle example from Mitrovic et al. (2003) and note, in addition to the constraint listed above, the third constraint.)

MO point to the following constraint in KWR (p. 122) as somehow specifying "correct answers or actions" in the satisfaction condition:

- C<sub>r</sub>     The population standard deviation is unknown, and  
          The sample size  $\geq 30$ , and  
          The student has input the test statistic to be used ("z" or "t").
- C<sub>s</sub>     The value entered for the test statistic to be used is "z".

In fact, this constraint captures a domain principle in statistical hypothesis testing: under the conditions specified in the relevance conditions, the correct value for the test statistic is "z". It perhaps would have been more conventional to state:

- C<sub>s</sub>     The test statistic is "z".

In any case, the phrasing of the English translation of the constraint is irrelevant; our implementation specifies constraints as domain principles in the same way constraints presented in the CBM literature do.

Perhaps a simpler constraint in our system that is conceptually equivalent, in terms of representing a domain principle, will make this clear to readers unfamiliar with statistical hypothesis testing.

- C<sub>r</sub>     The data set is known,  
          The student has calculated the sample mean,  $\bar{x}$

- C<sub>s</sub>     The value for the sample mean,  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$

Thus there is no difference in how KWR have encoded their constraints and how CBM recommends that it be done. Because constraints encode domain principles, a constraint violation either implicitly or explicitly specifies a "correct answer" to some part of the problem. The answer field in KWR's constraint template is used when a constraint violation can be traced to a single incorrect student variable. In that case, the system provides the student with an option to view the correct answer, which the tutor arrives at by evaluating the generic answer template in the context of the specific problem. This is analogous to the way CBM tutors in the literature use the ideal solution as a variable in their constraints.

MO state (p. 278):

"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."

This too sounds like a serious claim, but the statement is vague, and as we have already demonstrated, the constraints in KWR are no different from those in the CBM literature. Yet again, MO fail to connect their criticism with any conclusions drawn in KWR.

KWR (p. 119) indicate that in MTTs there are "rules for decomposing a problem into subproblems (or 'planning' rules) and rules that address the solution of atomic subproblems ('operator' or 'execution' rules)," and show that there is a one-to-one correspondence between operator rules and

constraints that capture domain principles (what KWR refer to as operator constraints). KWR then contend that a "pure" CMBT contains only operator constraints.

MO state (p. 278):

"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."

A short digression is in order here. It appears that MO are indicating that in "living tutors" not all constraints capture general principles of the domain. This is major news indeed; we would expect this caveat to appear prominently in the CBM literature – it doesn't. Further we would expect that in any publication describing the development and/or evaluation of a CBM, the number of constraints that do not represent domain principles would be specified along with the total number of constraints. We have never seen this number reported. More generally, we wonder why and under what conditions the CBM formalism breaks down such that it is needed to make this exception? MO don't say. Finally, what does "bulk" mean in the last sentence? Do 90% of the constraints in a CBMT have to encode principles of the domain? 60%? This issue raises important questions regarding the relative ease of building CBMTs, as well as the fundamental notion of CBM that effective remediation can be provided by general domain principles alone. Beyond this, the CBM camp argues that an advantage of the paradigm relative to MTTs is that the bug libraries in MTTs "tend not to transfer". (This topic is addressed in more detail later in this paper.) The transferability of these non-domain-principle constraints may be uncertain as well.

In any case, as we've made clear, the operator constraints in KWR do indeed represent domain principles. Further, although MO claim fundamental differences between the rule and constraint formalisms, an example from Mitrovic et al. (2003) indicates otherwise. This example, from the geometry domain and cited earlier here, consists of three rules and a corresponding set of three constraints and serves to support KWR's assertion that there is really a very close association between the two formalisms; in fact there is a one-to-one correspondence between operator rules and (operator) constraints. (See Appendix A, part 1 for the example.) This is, in fact, not such a surprising result. Suppose the relevance condition of a certain constraint holds. The satisfaction condition is not going to magically hold; obviously one or more student actions are needed to make the problem state fulfill the satisfaction condition. These actions are nothing but applications of one or more operators in the problem domain. Therein lies the inescapable relationship noted in KWR between constraints and operator rules.

In summary, MO's claim that KWR did not represent constraints properly is invalid.

Consider now MO's main claim that a central tenet of CBM is that constraints should encode general domain principles. The example constraint that they provide on page 279 (constraint number 476) clearly violates this requirement, notwithstanding their explicit statement to the contrary, that "It describes a correct domain principle". We reproduce the constraint below:

(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")
```

Constraint 476 is used in SQL-Tutor to try to determine if, in a query that uses a JOIN operation, the correct number of joins is specified in relation to the number of tables.

Constraint 476 makes the assumption that if the FROM clause has  $n$  tables and the student has not used the JOIN keyword in the FROM clause, then there must be  $n-1$  join conditions in the WHERE clause. The relevant fragment lies in the last 3 lines of the constraint (these represent the satisfaction condition of the constraint). This might be a characteristic of the problems they used, but it is certainly not a general domain principle as there are situations where there are  $n$  tables and the number of join conditions need not be  $n-1$ . For example, consider the classic "Supplier-Part-Project-Shipment" situation (Date, 1995) and the query, "Retrieve all details of shipments in which the supplier, part and project are located in the same city". The answer to this uses 4 tables, but needs 5 join conditions. We discuss the problem and the solution in greater detail in Appendix B. The critical point is this: there is no general principle in the domain of SQL that relates the number of tables to the number of join conditions and therefore constraint 476 does not embody a general domain principle.

The fact that the constraint MO chose to showcase does not represent a domain principle but rather only holds for the set of problems SQL-Tutor uses raises at least two questions:

1. How faithful is the implementation of SQL-Tutor, the flagship tutor of the CBM fleet, to one of the canons of CBM, that constraints should represent domain principles?
2. How easy is it to write (or recognize) true constraints?

Buggy constraints: MO begin this section (p. 278) as follows: "Their decision to write constraints as if they were rules lead KWR to introduce something called 'buggy constraints'." This statement is invalid because, as we've made clear, KWR represent constraints exactly as specified in CBM literature.

KWR argue that using buggy constraints is "required for a CBMT to provide remediation equivalent to that of the corresponding MTT." MO respond (p. 284) that "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." KWR do not hold the remediation provided by MTTs as a gold standard to measure remediation provided by tutors. In trying to compare the remediation provided by our own tutors, we chose to rely on Ohlsson's (1994) concept of "pedagogically relevant equivalence classes". Suppose that we have two tutors for the same domain and two student behaviors that have different pedagogical significance. If one tutor treats the two student behaviors as the same for remediation purposes (classifies the behaviors into the same pedagogically relevant equivalence class), but the other tutor is able to distinguish between these two behaviors then it is reasonable to assume that the second tutor provides superior remediation.

MO then fallaciously reason that KWR's argument for buggy constraints "is contradicted by the fact that we [Mitrovic, Ohlsson and others] have implemented several constraint-based tutors, none of which contains any constraints similar to KWR's 'buggy constraints', and these tutors are effective." (p. 278-9). MO continue this argument and conclude that, "Constraints like the 'buggy' constraints in KWR's Figure 9 (p. 133) are not needed to provide remediation in a CBM tutor."

The reader should note that the claim KWR make about buggy constraints (that they are required for a CBMT to provide remediation equivalent to that of the corresponding MTT) and the claim that MO refute (that CBMTs can provide effective remediation without such constraints) are entirely different claims. By attempting to refute a weaker argument than KWR make, MO engage in the logical fallacy of a straw-man argument.

KWR provide ample support for their position. We can demonstrate the argument quite easily here. Consider a descriptive statistics CBMT that asks a student to determine the median value of a set of numbers. For simplicity, presume the number of numbers is odd. The correct procedure is to order the numbers, and then select the middle value. If the student provides the correct answer, the satisfaction condition (which specifies the expression for the correct value for the median) is not violated, and all is well. If the student provides an incorrect answer, the constraint will be violated and the system should provide the feedback associated with the constraint. However, without a buggy constraint all the system knows is that the answer provided by the student is not correct; it has no idea if the error resulted from the student providing the arithmetic mean instead of the median, not ordering the numbers before selecting the middle value, or is the result of some other mistake. (In fact, not ordering the data before picking the middle value is, in our experience, the error students most frequently make.) We demonstrate the same phenomenon in Appendix A, part 2 using the example provided in Mitrovic et al. (2003).

Without the buggy rule or buggy constraint, all wrong answers to the above problem are equivalent and the remediation provided for all wrong answers would be the same, even though the two solutions do not fall into the same instructionally relevant equivalence class, as mentioned above. It is in this sense that buggy rules (and buggy constraints) enable more focused remediation. MO claim (p. 279):

"...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."

Certainly a tutor without the buggy constraint can identify that the student's answer is wrong and provide *some* remediation, but the above statement does not explain how a constraint-based tutor can give *equivalent* remediation without a buggy constraint.

Ohlsson (1994, p. 187) speculated about three potential disadvantages of CBM, the third one being, "... it might be that the state constraints provide the wrong type of abstraction. They might not partition student behaviors into pedagogically relevant equivalent classes after all." This is precisely what happens in the absence of buggy constraints in the situations we've discussed here and in KWR.

KWR provide four examples (p. 133-134) where buggy constraints would be required in a CBMT for equivalent remediation as that of an MTT (with buggy rules); one of these examples is from the SQL domain. KWR claim that it is a common mistake for students to have multiple tables in their

queries, but to omit the relevant join conditions, and that a buggy constraint to cover this situation might be appropriate. MO ignore three of the examples provided and attempt to dismiss the SQL example by demonstrating that a properly written constraint (number 476), that is, one that encapsulates domain principles, can do the job equally well. However, as we've demonstrated in the previous section, this constraint does not reflect a general domain principle – it holds only for the specific set of problems used in SQL-Tutor. It therefore remains to be seen whether a non-buggy, domain-principle constraint can actually handle the SQL situation that KWR refer to.

Lack of path constraints (process constraints): MO make two points here, both of which are incorrect.

MO begin by claiming that the following statement in KWR is "in error" (p. 280): 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."

KWR's statement is grounded in the CBM literature:

"The basic assumption is that diagnostic information is not hidden in the sequence of student's actions, but in the situation (or problem state) that the student arrived at" (Mitrovic, Mayo, Suraweera, Martin, 2001).

"Cognitive tutors faithfully model the procedures that should be learned, while constraint-based tutors represent just the states the student should satisfy, and ignore completely the path involved" (Martin, 2001).

"Constraint-based student modeling is based on the notion that the student can be described in terms of entities more abstract than particular solution paths or strategies" (Ohlsson, 1994).

"...the diagnostic information associated with a highly informative step does not reside in the step *per se*, but in the properties of the resulting problem state" (Ohlsson, 1994).

"In CBM, we are not interested in what the student has done, but in what state they are currently in. As long as the student never reaches a state that is known to be wrong, they are free to perform whatever actions they please" (Mitrovic et al., 2003).

"Because the constraints express properties of problem states rather than action sequences, they are necessarily silent about goal hierarchies, plans, weak methods, and other entities that figure so prominently in information processing analysis of cognitive strategies. Constraint-based models ignore the question of how behavior, correct or incorrect is generated" (Ohlsson, 1994).

Although it is clear that constraints can be written in such a way that process information can be captured – KWR make this point explicitly – the essence of CBM according to its proponents, including Mitrovic and Ohlsson, has for years been that process information is not needed for remediation. Indeed this is (was?) considered a strength of the paradigm. In using such constraints, KWR assert that a CBMT ends up mimicking an MTT. The term "path constraint" is thus an oxymoron, according to the CBM proponents themselves. Whereas MO (inaccurately) accuse KWR of

foisting a "rule-based" approach on constraints, they demonstrate very effectively the real way of writing constraints like rules – via process/path constraints!

KWR's MTT implementation has planning rules and these serve to achieve goal decomposition. This takes the form of "if the goal is to achieve x, then set sub-goals to achieve y and z". As shown in KWR, (p. 130-131) it is possible to write constraints to capture planning, but then the spirit of CBM is lost. MO's claim – "This does not contradict the approach; it only widens the concept of a constraint to include path constraints as well as state constraints" (p. 280) is incompatible with the spirit of CBM as defined by its proponents.

Even in KWR's CBMT, the process/path constraints get very complex. It becomes very difficult to understand their meaning without a lot of effort. We shudder to imagine what these constraints would look like in a system with hundreds of constraints.

Finally, we are baffled that MO state, "KWR decided not to use path constraints in their own system." (p. 280). In fact it is clear that KWR have used such constraints. KWR point out that "it is however possible to write constraints in such a way that they capture planning knowledge" (p. 130, bottom). KWR explicitly state (p. 131) "More generally, the presence or absence of values for some variables helps to infer something about the process, and we have a set of constraints in our system that exploits this." The last paragraph on page 131 of KWR also discusses this matter in detail. The point is not whether writing such constraints is possible – it clearly is – but whether such constraints fall into the basic spirit of CBM.

MO speak of path constraints and the procedural CBMT NORMIT with great confidence (p. 280):

"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. ... 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."

It should be noted that nowhere in the NORMIT literature, or any of the other CBMT literature as far as we can tell, is the term "path constraint" used. Indeed not only do none of the references cited here by MO (Mitrovic, 2002, 2003, 2005; Mitrovic, Koedinger, & Martin, 2003) use the term "path constraint", none of them so much as uses the word "path". We raise this not as a petty academic quibble, but because MO give the impression that path constraints are an accepted concept/term in the CBM literature, thereby implying that such constraints, and dealing with procedural domains more generally, have been dealt with in a meaningful way by the CBMT camp. The reality is that before NORMIT, the CBM literature relegated the topic of procedural applications to minor, rare, speculation. And in the published reports of NORMIT, the mechanism by which procedure is taken into account is only vaguely, if at all, specified. Further, saying that NORMIT shows that CBMT can deal with a procedural domain may be technically true, but it is misleading. (See the following paragraph.) Finally, the first sentence of this excerpt from MO indicates that path constraints have been used to "great advantage" in systems (emphasis ours). MO only cite one (NORMIT) and we are unaware of any others.

NORMIT works in the domain of database normalization. The implementers chose to partition the solution process into six, fixed, sequential steps and force the student to work through the steps in order. Thus the procedural aspect of the problem is trivial to implement, via planning/path constraints or some other mechanism. (Indeed once the problem has been so partitioned, there is no need for path

constraints, and in fact the NORMIT literature referenced by MO never specifies how the sequencing was actually implemented.) Beyond this, at no point in the NORMIT literature cited by MO is there any speculation as to how the NORMIT approach to incorporating procedures into the CBM paradigm might work, or be extended, for "real" sequential applications – that is, those with multiple possible correct solution paths (like statistical hypothesis testing, or physics mechanics problems). As we state above, planning/path constraints can handle this in principle, but become unwieldy as the number of possible correct paths becomes large, and then there's still the issue of adherence to one of the core ideas of CBM to address.

Explicit problem types: MO state (p. 281):

"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 from a non-existent problem with a move that limited its usefulness."

KWR implemented their CBMTs via their generalized constraint-based-tutor generator VersaTutor (Kodaganallur, Weitz, & Rosenthal 2004, 2006). VersaTutor stores multiple tutors from possibly vastly different domains in it. The screen shot in Figure 2 of Kodaganallur et al. (2004) actually shows problems from widely differing problem domains in the left hand pane. The implementation of VersaTutor is such that the constraints for all tutors are stored together in a single relational table. The field ProblemType was used simply to designate which constraints apply to which domains. A different way to achieve this partitioning of constraints would have been to check the problem type of every constraint – this is what was meant by "achieving this through relevance conditions would have been too inefficient." When the student is working on a hypothesis testing problem, where is the need for the system to consider constraints for Java syntax problems?

Beyond this, in the hypothesis testing problem that KWR consider, there are several sub-types of hypothesis testing problems. Some constraints apply to all problems and some apply only to certain sub-types. For example, some constraints are relevant only to problems where the null hypothesis takes the form " $\mu \leq \mu_0$ ". In this case, the student should compute the right-side cutoff value. The constraint associated with the right-side cutoff value is not relevant to other sub-types, for example those that need only the left-side cutoff value to be computed.

Like SQL-Tutor and other CBM tutors, KWR's tutor is incapable of understanding natural language. It cannot fathom the form of the null hypothesis by parsing the problem text. Therefore, information on problem data that a student would pick up while reading the problem is given a-priori to the tutor. One such piece of information for the hypothesis testing problem is a label that indicates the form that the null hypothesis takes (eight of the 43 constraints in the system use this in the relevance condition). This is consistent with standard CBM practice.

A simple example is provided by the constraint from Mitrovic et al. (2003), included in Appendix A here:

Cr: A base angle of an isosceles triangle is known ( $\theta_1$ )

And the student has calculated the size of the other base angle ( $\theta_2$ )

Cs: The size of  $\theta_2$  is  $\theta_1$

This constraint applies only to an isosceles triangle, and the relevance condition checks to see if this property of the problem is satisfied.

We can also consider constraint 476 provided by MO: the sixth and seventh lines of this constraint have the relevance condition check whether the ideal solution to the problem has certain characteristics (the ideal solution either contains some JOIN conditions in the FROM clause or the WHERE clause). In other words, this constraint applies only to problems that have specific characteristics (as evidenced by properties of their ideal solutions). The creators of SQL-Tutor did not find a need to assign a label to these problem types, but it was appropriate for the hypothesis testing domain. What KWR did is in accordance with the CBMT literature.

Extraneous constraints: Based on their implementation experience, KWR assert (p. 132), "It appears that, unless writing constraints is influenced by process considerations, it is possible that unnecessary effort might be expended in creating pedagogically insignificant constraints – what we have termed "extraneous consistency constraints".

MO respond first that, "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" (p. 281). In our view, this was not a "sweeping conclusion", but rather an observation that we felt had merit. We leave it to the reader to decide for themselves which of the two characterizations apply. We also leave it to the reader to decide whether the tone of MO's reply here, and elsewhere, is appropriate.

From MO (p. 281):

"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 [i.e.  $z$  and  $t$ ] 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 statistics 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."

This discussion raises a very interesting point. One of us, who has taught statistics for years in several institutions, has never had a student choose two different statistics for the same hypothesis testing problem. An informal canvassing of colleagues who teach statistics revealed the same results. However, MO are correct in stating that this may be viewed as a domain principle. The question raised by KWR is, when do you stop? Does this mean that in a physics tutor, a reasonable constraint to include would be one that specifies that an object (with respect to the same reference coordinate) can't be moving and not moving at the same time? After all this is a domain principle and it's conceivable that "some student, somewhere, will no doubt make this error." MO note (p. 281) that "nothing prevents the designer of a CBM system from thinking about students in terms of cognitive processes." We wonder whose cognitive processes? The tutor writer's? Students the tutor writer knows? This seems a far cry from the promise of simply writing constraints that "capture domain knowledge". More generally we note that writing constraints may not be such an easy proposition; we return to this idea later in a different context.

MO recap their critique to this point as follows (p. 281):

"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."

We have refuted each of these points.

### Range of applicability

Figure 10 of KWR (reproduced below) summarizes their argument about the range of applicability of the two paradigms.

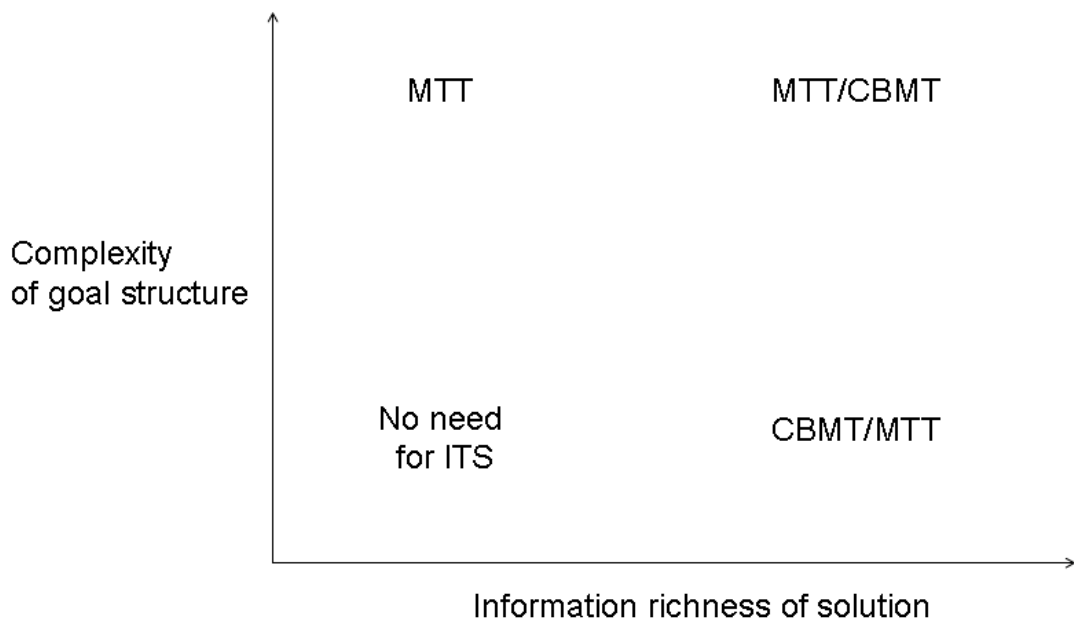


Fig. 1. KWR's Figure 10 on the range of applicability of the two paradigms.

On page 139, KWR clearly state:

"We have identified two dimensions that enable us to characterize the problem domains for which the two paradigms (**in their pure forms**) are suited." (Emphasis added here).

KWR have also stated explicitly that a CBM that relies on process constraints can mimic an MTT by subsuming into the problem state the products of intermediate steps. This discussion assumes that this has not been done.

KWR state that CBM is disadvantaged under two situations a) when the goal structure in the task is complex and b) when the solution has low information content. MO disagree with both of these contentions. We address each in turn.

Goal structure in a problem represents how the main goal is recursively broken down into sub-goals, as well as the ordering relationships between the various sub-goals. (Figure 1 from KWR shows an example.) The complexity of a goal structure increases with the number of levels, branching factor and sub-goal ordering restrictions.

When the goal structure for a problem becomes complex, it is intuitively clear that planning advice becomes critical. McKendree's (1990) findings stress the importance of planning advice. In a problem with a complex goal structure, a good deal of the learning involves understanding this structure, which is addressed by planning remediation.

As already discussed, a CBMT is not capable of providing planning advice unless it has an associated solver. As KWR have shown (without being challenged by MO), "The whole area of constraint-based solvers is in a nascent stage and would have to advance significantly in order to enable CBMT to match the planning remediation available in MTT." (p. 135). It is in this sense that CBM is at a disadvantage when the goal structure becomes complex.

Beyond the planning limitations, KWR argue that CBM has difficulty fathoming where the student is in the problem-solving process when the goal structure is complex. MO reply as follows (p. 282):

"...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 behavior 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."

It is not clear what MO mean by "detects a student error immediately after any step" or "monitored continuously". In SQL Tutor and in NORMIT, for example, the constraint-checking process only begins when the student explicitly submits a solution – in this sense there is no continuous monitoring. "Continuous monitoring" could mean keystroke level monitoring, or at the very least, that as soon as the student completes working on a field it is checked by the tutor.

In fact, the "monitoring" process is described quite clearly in the CBM literature. "Unlike ITSs that use model tracing ... constraint-based tutors do not follow each student's solution step-by-step: a student's solution is only evaluated once it is submitted, although the student may submit a partial solution to get ideas on how to progress" (Mitrovic, Suraweera, Martin, & Weerasinghe, 2004, p. 413).

If, what MO mean here is that a CBMT will accept and provide remediation for partial solutions (when the student submits one), they should simply say so. However, in our experience, submitting a partial solution to the SQL-Tutor is problematic. That is, contrary to MO's statement above, it in fact is difficult for SQL-Tutor "to fathom where in the problem-solving process the student has made an error."

In SQL-Tutor the user interface allows the student to enter a complete SQL query by filling in various individual fields. It is conceivable that a student might simply supply a value for one of the

fields (for example, just the table name) and submit the solution for evaluation. In this case, a tutor that knew where the student was in the problem solving process would not point out errors in the student solution that arise simply because of the incompleteness. It should be able to just check the correctness of what has been submitted. SQL-Tutor is in fact unaware of where the student is in the above situation. On problem 31, when the table name alone is provided, it actually spots 6 errors (see Figure 2 below). Many of these result from checking for constraints concerned with mismatches between the table name and other non-existent clauses in the student's solution – the tutor is not aware that the student has only worked on the table name and on nothing else yet.

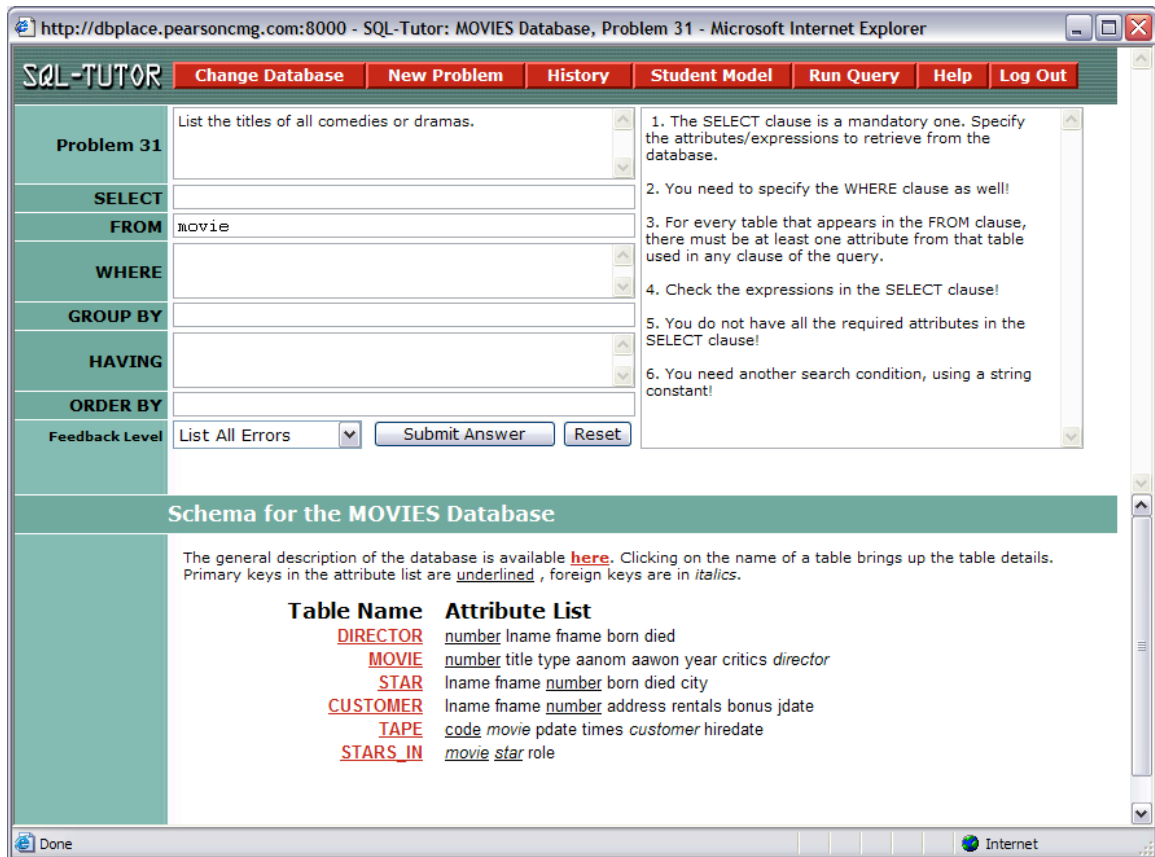


Fig. 2. Sample interaction with SQL-Tutor in which a partial, correct solution results in six error messages.

Of course a CBMT could break the solution process down into steps that may be remediated individually using process/path constraints. However the argument made in KWR explicitly applies to CBMTs that are built on domain principles and not those that merely mimic the procedural abilities of MTTs.

The second important determinant of the suitability of a paradigm for a domain is the information richness of the solution state – all the information that the tutor has available on which to base remediation. CBM, being state based, can perform well when this state is information rich because the state by itself reveals a lot about the student's mental model. However, CBM is hindered when the

problem state is information impoverished – there is not much information on which to base remediation.

For a concrete example, consider a problem in which the student is required to compute the value of the expression  $5 + 4x$ , given that the value of  $x$  is 6. Here the goal might be to teach operator precedence (that, in the absence of parentheses, multiplication precedes addition). The answer that the student provides is just a simple number – without much information content. Suppose the student provides the number 54 as the answer (most likely because of applying the addition operator before the multiplication operator). A model-tracing tutor, being interested in the *process* that the student adopted can infer the mistake via an explicit rule that can trace the action. Based on the available information, a constraint-based tutor can only recognize that the solution is wrong, but not the specific reason - its student model is weaker.

Indeed the purpose of the instructions on typical mathematics, physics and chemistry exams to "show all your work" is to allow instructors to give partial credit when a student submits an incorrect answer. If the (incorrect) answer alone gave sufficient information for the instructor to divine where the student went wrong (that is, what the student knows and doesn't know), these instructions would hardly be necessary.

It appears that Ohlsson (1994) was concerned with this possibility when he mused about the potential disadvantages in the CBM paradigm (p. 187): "... for some domains it might be impossible to identify properties of problem states which [*sic*] are highly informative with respect to the student's understanding."

MO sidestep the other examples in KWR (p. 139) and attempt to refute KWR, by stating:

"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." (p. 282)

In CAPIT, claiming that the results of the revisions do not have any internal structure because a capital letter has no meaningful internal structure is misleading – the complete text with the capitalized letter(s) and /or punctuation(s) in place is available – not just a single capital letter or punctuation mark. This is ample information on which to base remediation for this simple application. We note additionally that the simplicity of the application is discussed in Martin (2001, p. 45, bottom). The user interface of CAPIT (from Mayo & Mitrovic, 2001) is shown below (Figure 3) to reinforce the fact that the complete sentence which provides the context for the student's work is available to the tutor.

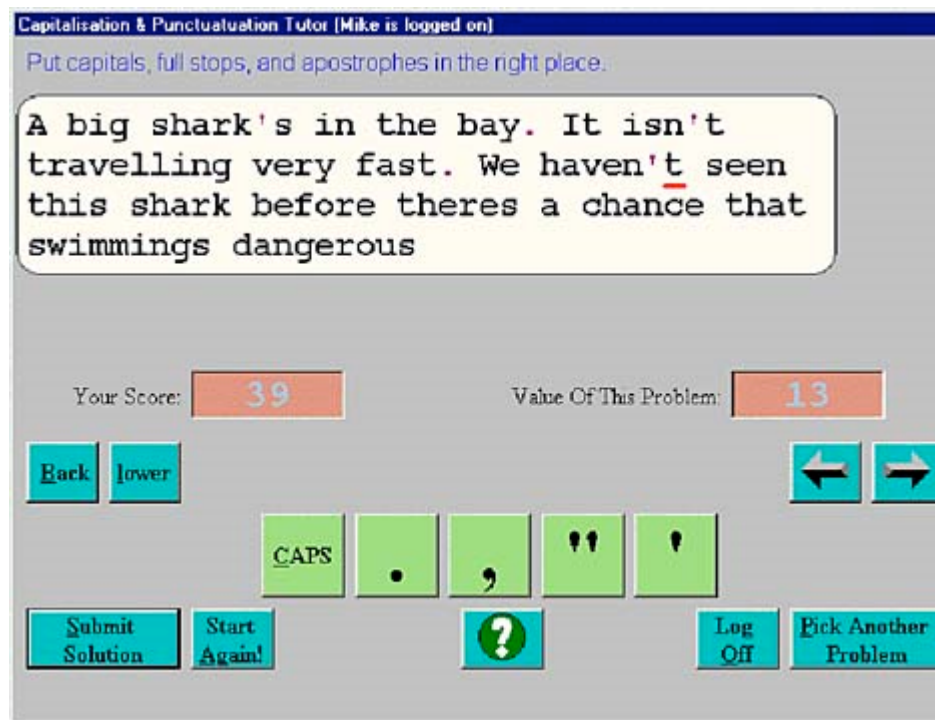


Fig.3. A screen shot from the CAPIT CBMT (from Mayo & Mitrovic, 2001).

We note that in the case when the goal structure is not complex, but the solution state is information rich, MTTs can still be constructed, but, since planning advice is not critical, or not even needed, their knowledge bases will consist of operator rules and buggy rules. In these situations it might be more natural to construct constraint-based tutors.

MO claim (p. 283):

"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."

To begin with, it should be noted that the CBM literature appears less convinced about the impossibility of implementing SQL and database design MTTs than MO indicate. Consider the following from Martin (2001, p. 28): "There is no right or wrong way to approach writing an SQL query. ... it is possible to encode all variations as separate paths through the production rule model, but this would make the mode [sic] large and unwieldy." A more recent source, Mitrovic, Suraweera, Martin and Weerasinghe (2004, p. 412) state, "CBM does not require an executable domain model,

and is applicable in situations in which such a model would be difficult to construct (such as database design or SQL query generation)."

So it appears that an MTT approach for SQL and database design has been, at minimum, considered feasible. In any case, MO's current claim that both are not possible domains for MTTs is worth exploring.

Suraweera and Mitrovic (2004) have a whole section devoted to "CBM and design tasks". According to them, design tasks are generally under-specified, are often open-ended or at least have a very large number of solutions, and there are no well-defined solution procedures for them.

The overall logic of the CBM camp is:

- Model-tracing tutors cannot be built for domains that lack well-defined problem solving procedures
- Design tasks do not have well-defined problem solving procedures
- **(Conclusion 1) - Therefore model-tracing tutors cannot be built for design tasks**
- SQL and database design have many characteristics of design tasks
- **(Conclusion 2) - Therefore model-tracing tutors cannot be built for SQL and database design**
- There exist CBM tutors for SQL and database design (SQL-Tutor and KERMIT)
- **(Conclusion 3) - Therefore CBM is applicable to design tasks**

The above logic seems impeccable, and would be acceptable if the actual tasks addressed by SQL-Tutor and KERMIT could be characterized as design tasks. The facts speak otherwise.

Mitrovic et al. (2003) performed their comparison of the paradigms based on a CBM and MT implementation for database design! They took a subset of the constraints of KERMIT (a fielded constraint-based tutor for database design) and implemented the corresponding set of rules. How could this be possible if there is no problem-solving procedure for database design? They do not mention the reason for taking only a subset of KERMIT's constraints. Surely if the reason was that it was impossible to represent the full-blown set of constraints as a set of rules due to inherent problems with the rule representation formalism, then this would certainly have been a crucial finding of their comparison effort. In fact they conclude their comparison on a very neutral note.

There is no doubt that, in general, the domains of SQL and database design have characteristics of design tasks. However, SQL-Tutor and KERMIT operate based on an ideal solution to each problem and can only provide remediation if the student solution is close to this ideal solution or – in the case of SQL-Tutor with the 'solver' – can be derived from the ideal solution. KERMIT deals with basic database design problems (of the type typically found in introductory college texts) in which there is very little scope for students to come up with alternative correct solutions. In other words, these tutors deliberately reduce task complexity to the point where they are not design tasks anymore. We hasten to add here that we do not doubt the usefulness of providing tutors for these restricted domains – they are indeed essential to teach the fundamentals of any domain. However, to make exaggerated claims based on these tutors is illogical at best and misleading at worst.

With the task complexity reduced so drastically, it is in fact quite easy to come up with expert solvers in these domains, and therefore model-tracing tutors **can** be built for them. This is exactly why in Mitrovic, Koedinger and Martin, (2003) they were able to create a model-tracing implementation of a CBMT for database design. There are in fact commercial products that actually generate SQL statements from natural language problem descriptions; see, for example, Microsoft (2006) and Hess

(1999). Indeed we can demonstrate that it is possible to come up with a set of production rules that can generate SQL statements. The details are provided in Appendix C.

## Remediation

We have already pointed out the following:

- MTTs outperform CBMTs on planning related remediation.
- Encoding only domain principles as constraints is not guaranteed to partition all instructionally distinct student behaviors into different classes for remediation. Buggy constraints might be needed to achieve this. A CBMT without such constraints would be at a disadvantage.

MO accuse KWR of misusing quotations, and they cite what they claim are several examples. One of them is that (p. 287) "Mitrovic et al. (2003) do not claim that remediation by CBM is necessarily or in general less comprehensive than remediation by MT..." In fact, Mitrovic et al. (2003) say exactly that (p. 321):

"Creating constraint-based modeling systems tends to require less time and effort, but the result tends to be less comprehensive in terms of specific advice-giving capabilities. Creating model-tracing tutors tends to require more time and effort, but this results in more specific advice-giving capabilities."

(The charge is particularly perplexing as Mitrovic is a co-author of both papers.) The other examples cited by MO are equally puzzling.

Based on this reference KWR claim (p. 139)

"As stated earlier it is generally accepted that MTTs outperform CBMTs with respect to remediation quality."

MO misquote the above (p. 287); they omit the first part of the sentence "As stated earlier," capitalize the 'i' in 'It', and then argue that KWR do not provide references to support the claim. The reference to Mitrovic et al. (2003) was provided earlier in KWR (p. 132, bottom). The fact that Mitrovic et al. (2003) was co-authored by a leading figure from each of the two paradigms provides support for claiming "general" acceptance of MTT's superior remediation quality.

However, we have reflected on the statement citing this reference in KWR and feel that it is perhaps misleading. While the fact is KWR repeatedly indicate the differences in remediation capabilities of CBMTs and MTTs, and the leading figures in the field provide support, MO is correct that there is only one other source aside from KWR that compares the two approaches. Therefore, in the interest of absolute precision, we have suggested to the editor that the sentence, "As the CBMT literature concedes (Mitrovic et al., 2003) the remediation provided by a CBMT is never superior to that provided by an MTT." be changed to "As Mitrovic et al. (2003) indicate, the remediation provided by CBMTs tends to be 'less comprehensive' than that provided by MTTs." The revised text appears in the print version of the journal.

MO incorrectly claim (p. 284):

"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"** (emphasis ours).

This statement might be true of many CBMTs, but is certainly not true of SQL-Tutor. SQL-Tutor recognizes some incorrect solutions as correct and some correct solutions as incorrect. Appendix D has more details. These failures were not the result of bugs, but arise from the fact that the ideal solution that SQL-Tutor stores adopted a different approach than that of the student as well as because SQL-Tutor does not recognize the equivalence of different ways of writing an expression. The correct and incorrect solutions we tried out were perfectly reasonable ones that typical students might provide – we were not trying to trick the system by giving devious solutions. One might counter that in a domain as complex as SQL, where there are many correct answers to any problem, it is unreasonable to expect the tutor to work perfectly under all conditions. Under these circumstances, MO should not have made such a bold claim. More importantly, the fact that the system sometimes flags correct solutions as incorrect implies that the correct solution actually violates some constraint. If all constraints represent domain principles, how can this happen?

The point of the above discussion is that although it is an appealing idea that we can simply represent domain principles as constraints and be assured that no correct solutions will ever violate a constraint, doing this could be extremely difficult in practice. The fact that a system which has evolved over several years, and has over 700 constraints can be tripped up by highly plausible correct and incorrect solutions is ample evidence of the difficulty of implementing the elegant promise of CBM.

KWR point out that CBMTs encounter problems when a solution state violates several constraints, and give examples of small errors (like misspelling a table name) that result in several constraint violations and corresponding messages. MTTs can also face problems when several rule sequences can successfully trace a single student solution. To this extent both types of tutors have similar problems. However, an MTT would break up the problem into steps based on the goal structure and the rules corresponding to higher order goals will not be triggered unless the corresponding lower order goals are satisfied. Thus even if the user interface provided the student with an option to make an error corresponding to a lower order goal and a corresponding higher order goal, the MTT would flag only the former error. In a CBMT, all constraint violations would typically be flagged and hence the user would see several error messages.

MO (p. 284-5) compare the nature of the remediation provided by CBMTs and MTTs as follows:

"Because CBM catches errors via constraint violations, a CBM tutor in which the constraints encode domain principles can talk to students about those principles. ... 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."

In fact, VanLehn et al. (2005), in describing the rationale for the Andes physics MTT, state (p. 157), "If students are going to solve problems, then they must master all principles, both major and minor. This is the first instructional objective of Andes." One conclusion of the evaluation studies they reported is that (p. 188), "Andes students scored higher than Control students on drawing and variable usage, which are two fundamental conceptual skills emphasized by Andes."

Further, McKendree (1990) found:

"...feedback about the goal structure of geometry problems led to better performance than feedback about the reasons for error or simply being told that an error has occurred. This goal feedback allows students to correct the incorrect action more often than other types of feedback."

Finally, in summary, in discussing the Andes physics MTT, VanLehn et al. (2005, p. 157) note:

"All physics problem solving systems have knowledge bases that include principles, and in fact, they tend to agree on what those principles are, even the minor ones. However, if such a system knows only principles, then it may produce a huge list of equations, most of which are unnecessary, as it solves a problem. Experts and competent students clearly have some other knowledge than principles, and they use this knowledge to constrain their reasoning so that they produce just the principle applications required for solving the problem."

We have made a similar observation in our detailed examination of CBM in general and SQL-Tutor in particular. The notion that high-quality remediation can be provided based on general principles is a seductive one. Our opinion is, in practice there are significant limitations to this approach. We believe we have amply demonstrated the major problems.

## **Methodology**

We address each of the broad criticisms of our methodology in turn.

Unwarranted generalizations: This section of MO is a rehash of previous issues that have already been addressed.

Incomplete comparisons: There are two issues raised by MO that we address here. First, MO state (p. 285-6):

"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....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"

In fact, KWR state (p. 132): "One of the advantages claimed for CBMTs is that they do not require extensive bug libraries (Ohlsson, 1994, Mitrovic et al., 2001) and are hence less resource intensive to build." More generally KWR indicate (p. 141):

"Our experience and analysis indicate that building a pure CBMT might be easier than building an MTT for the same domain because of the extra effort needed to develop expert and buggy planning rules and buggy operator rules for MTT. We also found that building the MTT rule set was a somewhat more complex exercise because of the need to tie the rules together through a goal structure. .... In sum, our observations lend support to CBMT proponents who argue that building CBMTs is less resource intensive."

Second, MO dismiss the examples provided by KWR of poor remediation afforded by SQL-Tutor as the kind of simple bugs expected in any software. KWR argued that these instances represented fundamental problems with SQL-Tutor and reflected the real difficulties they noted with CBM. As stated in the previous section, we provide additional such examples and a more detailed analysis in Appendix D.

We have already addressed the issues raised by MO in the last paragraph of this subsection.

Use of the Literature: MO present a shopping list of claims regarding alleged misrepresentation of the literature by KWR. We encourage interested readers to check these claims for themselves. In any case, we address each of them here.

MO state that (p. 286):

"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."

KWR never claim CBM cannot be applied to procedural domains – in fact KWR demonstrate quite clearly, and we've reiterated here, that it can. MO don't specify which of the "many" other arguments in KWR would have been disproved had NORMIT been included. The truth is that there is nothing in NORMIT that invalidates anything in KWR, as we have shown in quite some detail here.

MO continue, "KWR do not cite our journal article about KERMIT (Suraweera & Mitrovic, 2004), although this paper was available before KWR's paper was published."

The KERMIT paper was indeed available before KWR was published; however what is relevant is that the KERMIT paper did not appear until after we submitted KWR. Beyond this, it is immaterial which occurred first, as KWR do cite the earlier conference paper describing KERMIT, and the journal article is just an expanded version of the conference paper that does not affect KWR's argument in any way. MO go on to state that KWR cite an early conference paper on WETAS instead of later papers; this overlooks the fact that the issue of ITS development environments is largely tangential to the KWR paper. We didn't mention various MTT papers either – including those describing the latest advances in MTT development environments.

MO state (p. 287):

"KWR support their claim that an MT tutor can be built for every domain in which a CBM tutor can be built 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' (p. 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."

KWR are quite clear on this point. KWR argue that an MTT can be developed for any domain that a constraint-based tutor supports. KWR do not claim that Mitrovic et al. (2003) say this; KWR

only say that the example Mitrovic et al. (2003) provide is in line with this statement – that is, they built an MTT for an existing CBMT and not the other way around.

MO continue:

"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" (see 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."

Let's take these one at a time.

The charge in MO: "Mitrovic et al. (2003) do not claim that remediation by CBM is necessarily or in general less comprehensive than remediation by MT..." is false and has already been addressed here in the Remediation section.

The charge in MO: "Martin (2001) did not argue that CBM systems have a high probability of providing misleading remediation."

Here's the relevant excerpt from KWR (p. 135):

"Since CBMTs are not designed to fathom the strategy that a student is adopting to solve a problem, they have been known to give misleading remediation when the student has provided an unexpected, but correct, solution... Martin's dissertation (2001) reports on this problem in detail and outlines the extensive work needed to be done in order to improve remediation in this scenario."

The reader should note that KWR say CBMTs "*have been known to* give misleading remediation," not that they "*have a high probability of* providing misleading remediation." KWR refers here to section 3.1 of Martin (2001), entitled, "Feedback Can Be Misleading" in which some challenges of providing accurate remediation under the CBM paradigm are forthrightly discussed.

Finally, according to MO (p. 287), "Mitrovic et al. (2003) do not claim that ... CBM ... necessarily compromises cognitive fidelity."

Mitrovic et al. (2003) addresses the issue of cognitive fidelity in two places. Table 2 (p. 319) summarizes the differences discussed in the paper between the two approaches; the second line highlights cognitive fidelity and the entry for MT is "tends to be higher" and the entry for CBM is "tends to be lower". In the discussion below this table, is the following: "MT tutors represent domain knowledge as production rules. This knowledge has high cognitive fidelity, because it is an explicit model of the reasoning that the learner must acquire. High cognitive fidelity is a strong advantage of Cognitive Tutors."

However, we do want to clarify a statement in this regard made in KWR (p. 139). The relevant section is:

"We have seen that it is possible to build a CBMT that emulates an MTT by incorporating planning and buggy constraints. However, this violates the CBM

paradigm. In addition, as Mitrovic et al. (2003) specify, this compromises cognitive fidelity."

As we've noted above, Mitrovic et al. (2003) indicates that CBMTs tend to have lower cognitive fidelity than MTTs *generally*, not as a result of including path/process constraints to emulate the procedural approach taken by MTTs. It is our belief that adding process/path constraints can only exacerbate the cognitive fidelity gap between MT and CBM but, to be clear, that is our belief, and not one expressed in Mitrovic et al. (2003).

## **MO's Conclusion**

MO conclude by suggesting that the right way to compare the two paradigms is via empirical comparison studies. They cite a litany of difficulties in performing such studies and suggest in the meantime, "For system builders, it is not a trivial question whether a particular design philosophy makes them do more or less work."

They then repeat their argument that CBMTs don't need buggy rules and that this is a significant advantage over MTTs (p. 288): "Moreover, what little evidence there is on the matter suggest 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."

(They also take KWR to task again here for not mentioning this, which as we've pointed out previously, it does.)

However, irrespective of any of the arguments presented in KWR or here regarding buggy rules/buggy constraints, the notion of non-transferability of buggy rules should not go unchallenged. The CBM literature previously expresses certainty regarding the non-transferability of bug rules:

Mitrovic et al. (2003): "Studies have shown that bug libraries do not transfer well to new population of students; if a bug library is developed for a certain group of students, it may not cover the bugs that another group of students may make [(Payne & Squibb, 1990)."]

Suraweera and Mitrovic (2004): "... bug libraries do not transfer well between different populations of students (Payne & Squibb, 1990)."

Mitrovic and Ohlsson (1999): "Another advantage of constraint-based student modeling over the bug library technique is that bug libraries do not transfer well between different populations of students (Payne & Squibb, 1990)."

Interestingly, MO restate this argument here with the preface "what little evidence on the matter suggests". Apparently the little evidence they are referring to is the single article cited in these three papers (and in Ohlsson, 1994). This study focused on algebra problem-solving of 13-14 year olds, in three different English secondary schools. While it appears to be a thoughtful and well-conducted experiment, it seems an unmerited generalization to presume bug libraries in any and all situations "don't transfer well". It would seem that additional studies are required before this wide-ranging claim regarding the non-transferability of buggy rules can be substantiated.

MO are quite correct in pointing out that conducting empirical comparisons of the two paradigms will be challenging to accomplish. We suggest in the meantime that they make their tutors available to

the general public so that interested researchers can freely evaluate the behavior and performance of CBMTs for themselves. (We note the Andes physics MTT is unrestrictedly available online (at <http://www.andes.pitt.edu/>), as is the Ms. Lindquist algebra tutor (at [www.algebratutor.org](http://www.algebratutor.org)).

## SUMMARY/CONCLUSIONS

KWR's work originated from a research project with the goal of developing an intelligent tutor in the domain of statistical hypothesis testing. As part of that project, we became interested in comparing the two approaches. We had no interest in anything other than a straightforward, comprehensive exploration of the salient properties of both paradigms.

A reading of the sole other CBM – MT comparison paper, Mitrovic et al. (2003), reveals that the MT and CBM implementations were written by different people (p. 318). That is, in reality, neither camp has made sufficient effort to learn the other's paradigm well enough to undertake a real implementation. In this sense KWR's deep knowledge of both paradigms makes them uniquely qualified to perform a comparison.

In terms of the claims made by MO, we feel we have addressed them fully and honestly. None of the significant charges have merit. We do feel that two sentences in KWR were poorly worded and we regret that; however neither of these is of any consequence. We provide a summary of the main charges made by MO and our responses at the outset of the paper. The contribution of this paper goes beyond simple rebuttal; it challenges some of the fundamental claims of the CBMT literature. We hope that, in addition to heat, this interchange has shed considerable light on some important issues in the field of intelligent tutoring systems.

## REFERENCES

- Date, C.J. (1995). *Introduction to Database Systems*. 6<sup>th</sup> ed, Addison Wesley.
- Hess, Ed. (1999). Using Speech Recognition with Microsoft English Query. Microsoft Internet Developer. Retrieved January 30, 2006 from <http://www.microsoft.com/mind/0699/equery/equery.asp>.
- Kodaganallur, V., Weitz, R., & Rosenthal, D. (2004). VersaTutor – An Architecture for a Constraint-Based Intelligent Tutor Generator. In *Proceedings of the 13th Annual World Wide Web Conference WWW2004* (pp. 474-475).
- 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, 117-144.
- Kodaganallur, V., Weitz, R., & Rosenthal, D. (2006). Tools for Building Intelligent Tutoring Systems. In *Proceedings of the Thirty-Ninth Annual Hawaii International Conference on System Sciences (HICSS)*, Kauai, Hawaii, January 4-7, 2006, IEEE Computer Society.
- Martin, B. (2001). *Intelligent Tutoring Systems: The Practical Implementation of Constraint-Based Modelling*. Ph.D. Thesis, University of Canterbury, New Zealand.
- Mayo, M., Mitrovic, A., & McKenzie, J. (2000). CAPIT: An intelligent tutoring system for capitalization and punctuation. In Kinshuk, C. Jesshope & T. Okamoto, (Eds.) *Proceedings of International Workshop for Advanced Learning Technology: Design and Development Issues* (pp 151-154). Los Alamitos, CA: IEEE Computer Society.
- Mayo, M., & Mitrovic, A. (2001). Optimising ITS Behaviour with Bayesian Networks and Decision Theory. *International Journal of Artificial Intelligence in Education* 12, 124-153.

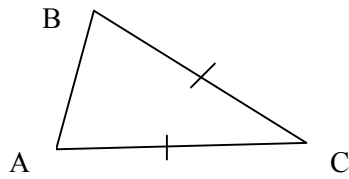
- McKendree, J. E. (1990). Effective feedback content for tutoring complex skills. *Human Computer Interaction*, 5, 381-414.
- Microsoft (2006). SQL Server 2000 Resource Kit, Chapter 32 - English Query Best Practices. Retrieved January 30, from <http://www.microsoft.com/technet/prodtechnol/sql/2000/reskit/part9/c3261.msp>.
- 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., Koedinger, K., & Martin, B. (2003). A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling. In P. Brusilovsky, A. Corbett & F. de Rosi (Eds.) *Proceedings of the 9th International Conference on User Modeling UM 2003* (pp. 313-322). LNAI 2702. Berlin: Springer-Verlag.
- Mitrovic, A., Mayo, M., Suraweera, P., & Martin, B. (2001). Constraint-Based Tutors: A Success Story. In L. Monostori, J. Vancza & M. Ali (Eds.) *Proceedings of the 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE – 2001)* (pp. 931-940). Berlin Heidelberg New York: Springer-Verlag.
- 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., & Ohlsson, S. (2006). A Critique of Kodaganallur, Weitz, and Rosenthal, "A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms". *International Journal of Artificial Intelligence in Education*, 16, 277-289.
- 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. (1994). Constraint-Based Student Modeling. In J. E. Greer & G. McCalla (Eds.) *Student Modeling: The Key to Individualized Knowledge-Based Instruction* (pp. 167-189).
- 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.
- VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., & Wintersgill, M. (2005). The Andes Physics Tutoring System: Lessons Learned. *International Journal of Artificial Intelligence in Education*, 15(3), 147-204.

## APPENDIX A

### Part 1

The following set of three MT rules and three constraints, which can be used for the same example of calculating angles of a triangle, is from Mitrovic et al. (2003).

The problem:



Angle A is  $65^\circ$   
What is angle C?

Two correct MT production rules:

IF The goal is to find an angle in an isosceles triangle ABC and  $AC = BC$  [Note: Mitrovic et al. (2003) misstated this equation as  $AC = AB$ ], and  
Angle A is known  
THEN Set the value of angle B to A

IF The goal is to find an angle in a triangle ABC and Angles A and B are known  
THEN Set the value of C to  $180 - A - B$

A buggy production rule:

IF The goal is to find an angle in an isosceles triangle ABC, and  
Angle A and C are at the bottom of the triangle and angle A is known  
THEN Set the value of angle C to A

Mitrovic et al. (2003) describes the buggy rule: "In geometry, students often over-generalize from common orientations of figures. This buggy production represents the shallow inference that the angles at the bottom of an isosceles triangle are always equal."

The associated constraints are:

- $C_{r1}$ : A base angle of an isosceles triangle is known ( $\theta_1$ )  
And the student has calculated the size of the other base angle  $\theta_2$
- $C_{s1}$ : The size of  $\theta_2$  is  $\theta_1$
- $C_{r2}$ : A base angle of an isosceles triangle is known ( $\theta_1$ )  
And the student has calculated that the size of another angle  $\theta_2$  that equals  $\theta_1$
- $C_{s2}$ :  $\theta_2$  is a base angle
- $C_{r3}$ : Two angles of a triangle are known ( $\theta_2$  and  $\theta_1$ ),  
And the student has calculated the size of the third angle  $\theta_3$
- $C_{s3}$ : The size of  $\theta_3$  is  $(180 - \theta_2 - \theta_1)$

The verbiage describing the constraints in Mitrovic et al. (2003), and the order in which the constraints are presented, in our opinion, cloud the direct connection between rules and constraints. However Mitrovic et al. (2003) clearly note, "The third constraint is equivalent to the second rule..." We add here that the same can be said about the first constraint and the first rule. This is a clear demonstration of the contention in KWR that there is a one-to-one relationship between operator (in the KWR sense) rules and constraints.

## Part 2

Mitrovic et al. (2003) contend, "Constraint 2 catches the same error as the buggy rule from Figure 1." Indeed the second constraint "catches the same error as the buggy rule" but it clearly cannot provide the same remediation. The aim of the buggy rule is to identify the error that the student has mistakenly set the value for angle C to that of angle A due to the orientation of the triangle. If a student violates constraint 2, all the CBMT knows is that the student has made an error in correctly identifying the second base angle – the system has no idea why the student made that error, and therefore cannot provide the same targeted feedback that the buggy rule can. This illustrates again the point made in KWR regarding the power of buggy constraints.

## APPENDIX B

Constraint 476 mentioned in MO is based on an assumption that if there are  $n$  tables in the FROM clause of an SQL query (which does not explicitly use the JOIN keyword in the FROM clause), then there must be  $n-1$  join conditions in the WHERE clause. This is not a general principle of the domain. There is no specific domain condition that restricts the number of join conditions in the WHERE clause. We give below an example in which the FROM clause has 4 tables, but the WHERE condition requires 5 join conditions. A join condition occurs when a field of one table is compared (usually, but not necessarily, for equality) with a field in a different table.

Date (1995) gives an example database schema that is reproduced below in slightly modified form for readability.

```
Supplier(Sno, sname, status, city)
Part(Pno, pname, pcolor, weight, pcity)
Project(Jno, jname, jcity)
Shipment(Sno, Pno, Jno, quantity)
```

The query "Retrieve all details of shipments where the supplier, part and project are in the same city" would be correctly solved by the SQL statement:

```
SELECT shipment.*
FROM supplier, part, project, shipment
WHERE
  (shipment.sno = supplier.sno) AND
  (shipment.pno = part.pno) AND
  (shipment.jno = project.jno) AND
  (scity = pcity) AND
  (pcity = jcity)
```

The above SQL statement joins the shipment tables with each of the tables supplier, part and project. In addition it has to ensure that the supplier, part and project are co-located and the last two join conditions achieve this.

Likewise it is perfectly possible (although very rare) to have no join conditions at all even though there is more than one table in the WHERE clause. This occurs when the Cartesian product of two (or more) tables is needed as output. For example (admittedly contrived), suppose a dress manufacturer

makes several types of pants and this information is stored in a table `Pant` and makes these in different sizes with the possible sizes stored in another table `Size`. Suppose we need all combinations of pant and size information to be listed, then the SQL query would be

```
SELECT *  
FROM Pant, Size
```

There is no need for a join condition because we want the Cartesian product to be listed.

## **APPENDIX C**

We provide here an argument demonstrating that it is possible to come up with a set of production rules that can generate SQL statements.

Given all problem-related information that a student will be able to glean from the natural language problem description, as well as information about the database schema, a set of production rules can be written to generate several alternate SQL queries by individually generating the various query elements and combining them. Examples of such a-priori elements would be the table name(s), field names needed, various clauses in the "where" condition, etc. The set of problems can be categorized into various subclasses (single table select without where condition, single table select with where condition, multiple table select with where conditions, etc.) and the specific nature of each problem can also be provided to the tutor up-front. With this information available, it is easy to write a set of production rules to generate several alternate correct and incorrect queries for each problem. In fact, production rules to arrive at several alternative correct solutions can be created and the system can then gracefully accommodate several correct solutions.

The big challenge for MTT in this case is that the number of incorrect student answers can also be huge because of the various ways in which the query components can be misspelled. An easy way to handle this is to convert the student answer into a canonical form in a pre-processing step, and then to trace this modified answer. For example, all field names in the `SELECT` clause and table names in the `FROM` could be alphabetically ordered. All individual elements of the where clause could be suitably ordered. All expressions could be converted into some standard canonical form (to handle the different ways in which the same arithmetic or logical expression could be expressed). All misspelled elements could be replaced by a common token, and so on. In this way it would be possible to convert entire groups of incorrect solutions into one, and have a single buggy rule to account for all incorrect solutions belonging to a single family.

## **APPENDIX D**

To illustrate the numerous significant problems we found with SQL-Tutor, we cite three of them below. We acknowledge that SQL is a complex domain and that expecting a tutor to work perfectly under all circumstances is impractical. However, the cases we report below are all simple ones that in our experience as database instructors we have encountered in practice. It is important to note that these have occurred in a mature tutor which has evolved over almost a decade and has over 700 constraints.

We would like to reiterate the point made in KWR: we did not spend a lot of time interacting with the tutor, nor did we attempt to fool it with devious responses.

What these errors point to is the difficulty of realizing the elegant promise of CBM: simply represent the domain principles as constraints and rest assured that a) a correct solution will not violate any constraint and will therefore be recognized as correct no matter what strategy the student adopts, and b) that an incorrect solution will violate some constraint and be found to be incorrect.

Example 1 - Problem 35: Incorrect solution flagged as correct.

Task: List the numbers and titles of all movies made between 1990 and 1993.

The stored ideal solution is

```
SELECT TITLE,NUMBER
FROM MOVIE
WHERE YEAR BETWEEN 1990 AND 1993
```

If the student does not use the BETWEEN clause, but instead breaks up the condition into two separate tests as YEAR >= 1990 AND YEAR <= 1993, the tutor still recognizes the solution as correct. However, the tutor also accepts the following incorrect solution as correct, apparently not being able to distinguish between the AND and OR logical operators in this context.

```
SELECT TITLE,NUMBER
FROM MOVIE
WHERE YEAR >= 1990 OR YEAR <= 1993
```

Example 2 - Problem 36: Correct solution flagged as incorrect

Task: List the numbers and titles of all science fiction ('sci fi') or horror movies.

The stored ideal solution is

```
SELECT NUMBER,TITLE
FROM MOVIE
WHERE TYPE IN ('sci fi','horror')
```

The following is also a correct solution, but the tutor flags it as incorrect.

```
SELECT NUMBER TITLE
FROM MOVIE
WHERE (TYPE = 'sci fi') OR (TYPE = 'horror')
```

The errors that the tutor finds are:

1. Make sure you have specified all the necessary conditions! Check the logical connective in WHERE!
2. Make sure you have specified all the necessary conditions! Check the logical connective in WHERE!

We verified that this solution gives the same results as the stored ideal solution when executed.

### Example 3 - Problem 64: Misleading remediation with unexpected, but correct solution

Task: Find the names and numbers of all directors who directed at least as many movies as director number 15.

The stored ideal solution is:

```
SELECT DIRECTOR.NUMBER, LNAME, FNAME
FROM DIRECTOR JOIN MOVIE ON DIRECTOR.NUMBER=DIRECTOR
GROUP BY DIRECTOR.NUMBER, LNAME, FNAME
HAVING COUNT(*) >= (SELECT COUNT(*) FROM MOVIE WHERE DIRECTOR=15)
```

Another correct solution that uses a different approach is given below. The tutor flags it as incorrect.

```
SELECT NUMBER, LNAME, FNAME
FROM DIRECTOR D1
WHERE
(SELECT COUNT(*) FROM MOVIE WHERE DIRECTOR = D1.NUMBER)
>=
(SELECT COUNT(*) FROM MOVIE WHERE DIRECTOR = 15)
```

The errors that SQL-Tutor lists in the above correct solution are:

1. Check that you use logical connectives (AND, OR) between conditions in the WHERE clause!
2. You need to specify the GROUP BY clause as well!
3. You need to restrict grouping --- specify the HAVING clause as well!
4. You need to use an integer constant in a condition in HAVING!
5. Do you need a search condition in WHERE, specified with a numerical constant?

Unfortunately the stored ideal solution when submitted for execution fails with the following error (probably because the underlying database system does not support the syntax used):

Database ERROR: [Microsoft][ODBC Microsoft Access Driver] Syntax error in FROM clause.

Our solution, on the other hand, executes perfectly and provides the expected results based on the database contents (we verified this manually).

The problem here is that the tutor is unable to cater to a solution that uses a different strategy from that which the stored ideal solution does. This correct solution apparently violates five constraints which represent universal domain principles! In reality what the solution violates is a set of constraints that perform shallow syntactic comparisons between the stored ideal solution and the student solution. It is a huge stretch to claim that these constraints represent general domain principles.