Int J Artif Intell Educ (2016) 26:457–473
DOI 10.1007/s40593-015-0075-7

ARTICLE

Constraint-Based Modeling: From Cognitive Theory to Computer Tutoring – and Back Again

Stellan Ohlsson [1]

Published online: 28 October 2015

Abstract The ideas behind the constraint-based modeling (CBM) approach to the design of intelligent tutoring systems (ITSs) grew out of attempts in the 1980's to clarify how declarative and procedural knowledge interact during skill acquisition. The learning theory that underpins CBM was based on two conceptual innovations. The first innovation was to represent declarative knowledge as constraints rather than chunks, propositions, or schemas. The second innovation was a cognitive mechanism that uses the information in constraint violations to revise and improve a partially mastered skill. This learning theory implied that an ITS could be built around a set of constraints that encode correct domain knowledge, without an explicit or generative model of buggy versions of a skill. Tutoring systems based on CBM have proven effective in multiple educational settings. CBM is limited in its focus on learning from errors. A broader learning theory, the Multiple Modes Theory, is outlined, and its implications for the design of more powerful ITSs are discussed.

Keywords Bug · Buggy model · Cognitive model · Constraint · Constraint-based model · Declarative knowledge · Learning mechanism · Learning by rote · Learning from error · Learning with understanding · Procedural knowledge · Skill acquisition · Tutoring

Introduction

The ideas behind the constraint-based modeling (CBM) approach to the design of intelligent tutoring systems grew out of attempts in the 1980's to clarify what it means to learn and execute a cognitive skill with understanding, as opposed to 'mechanically' or by 'rote'. School children who make nonsensical errors in mathematics are

✉ Stellan Ohlsson
stellan@uic.edu

[1] Department of Psychology, University of Illinois at Chicago, Behavioral Sciences Building, M/C 285, 1007 Harrison Street, Chicago, IL 60607, USA

prototypical examples of the latter (Hiebert, 1986). The learning theory that underpins CBM, *constraint-based rule specialization*, was invented to explain how declarative knowledge can help a student learn from his or her errors in the course of skill acquisition. Two key steps were to represent declarative knowledge in constraints rather than chunks, propositions, or schemas, and to invent a cognitive mechanism that uses constraints to detect and correct errors during skill practice (Ohlsson, 1993, 1996a, 1996b; Ohlsson & Rees, 1991a, 1991b). This learning theory implied that an intelligent tutoring system (ITS) could be built around a knowledge base of constraints that encode correct domain knowledge, without an explicit or generative model of students' buggy skills and hence without the need for labor intensive empirical studies to identify the latter (Ohlsson, 1992). The first CBM system was described in Mitrović and Ohlsson (1999).

In the following, I describe how the problem of learning cognitive skills with understanding was conceptualized twenty-five years ago, how it impacted the emerging ITS technology via the concept of buggy student models, and how the constraint-based approach filled two holes in the research literature. I refer the reader to Mitrović and Ohlsson (1999) and Mitrović (2012) for the subsequent story of how the CBM approach came to be implemented in tutoring systems that have benefited thousands of students worldwide. In the last section, I identify the main limitations of the CBM approach, sketch a new learning theory with a broader scope called the Nine Modes Theory, and discuss its implications for the design of more powerful tutoring systems.

But…But…But 324 Minus 65 *Cannot* Be Equal to 341!

The constraint-based theory of learning from error was the product of a research program at the Learning Research and Development Center (LRDC), an educational research center associated with the University of Pittsburgh. In the 1970s and 1980s, there was great enthusiasm in both psychology and education for applying the concepts and techniques of cognitive science to problems of instruction.[1] Pioneers like Richard Anderson, Carl Bereiter, John Seely Brown, Allan Collins, Robert Glaser, James Greeno, Walter Kintsch, Alan Lesgold, David Perkins, Lauren Resnick, Roger Schank, Derek Sleeman, James Voss, and many others inspired younger, second-generation cognitive scientists like myself to seek interesting research problems in the classroom, as opposed to the psychological laboratory.

A key cause of frustration for educators and researchers alike was the prevalence of errors in children's arithmetic. (At the time, the ability to do arithmetic correctly with paper and pencil was still considered a worthwhile pedagogical goal.) Teachers tore their hair and researchers scratched their heads over the persistent, recurring errors of commission. The fact that drove everyone to despair was that many of the errors seemed utterly unnecessary. Even a rudimentary understanding of what it means to subtract ought tell a child that the result of subtracting, for example, 65 from 324 cannot possibly be 341. In the world of the natural numbers, to subtract is to decrease, so how could a child believe that the result can be greater than the minuend?

[1] For example, see Lesgold, Pellegrino, Fokkema, and Glaser (1978); Klahr (1976), and Dillon and Sternberg (1986). Carver and Klahr (2001) provide retrospective reflections on the era.

Common sense suggests that such errors occur because school children fail to pay attention or lack the motivation to learn. But the dominant hypothesis among educators was that bizarre errors occur because children do not understand the meaning and the purpose of the arithmetic procedures they are asked to learn and execute, so they have no basis for judging whether their arithmetic operations make sense. This is a long-standing hypothesis. Brownell (1947) wrote that, "Error studies, for example, [have] disclosed faulty procedures which were [sic] explicable only as the results of blind groping on the part of children." (p. 259). Forty years later, Silver (1986) wrote, "Systematic bugs in procedures can often be traced to flaws in conceptual knowledge or to the lack of conceptual/procedural knowledge linkages." (p. 187) However, the despair of educators deepened when it was discovered that in some cases children could accurately judge someone else's arithmetic performance as correct or incorrect, indicating conceptual understanding of the relevant arithmetic procedure, but they nevertheless made errors of their own when asked to *perform* the same arithmetic procedure (e.g., Gelman & Meck, 1983). Why was the students' conceptual understanding activated in one context but not in the other?

Errors in school mathematics seemed an ideal research topic for those who wanted to apply cognitive concepts and techniques to education. Here was a central pedagogical problem, rooted in 'real life' and of indubitable societal relevance and importance; clear and circumscribed tasks requiring well-specified cognitive skills of varying complexity; and unambiguous, agreed-upon tests of mastery. To identify the relevant cognitive processes, explain the origin of errors, and develop instructional techniques that allowed students to bring their conceptual understanding to bear on their own performance and thus to avoid making bizarre errors seemed to be a research enterprise with a high probability of success.

Cognitive scientists brought several conceptual tools to bear on the problem. The distinction between *declarative* and *procedural* knowledge originally emerged in the field of Artificial Intelligence, specifically in the discussion of Terry Winograd's natural language understanding system *Shrdlu*, in which the meanings of sentences were represented in terms of the actions they implied, instead of the assertions they made (Winograd, 1972). Once formulated, the distinction was rejected by Artificial Intelligence researches themselves (Winograd, 1975), but it migrated into psychology (Chi & Ohlsson, 2005; Ohlsson, 1994), in part because John R. Anderson incorporated it into his soon-to-be-famous ACT model of cognition (Anderson, 1976, pp. 116–122). The distinction was useful in the discussion of children's mathematical errors because it made the co-existence of correct explanations and incorrect answers less mysterious. If there are two kinds of knowledge rather than one, then the two might grow at different rates, so it becomes understandable that correct (declarative) knowledge can co-exist in the learner's head alongside (incorrect) procedural knowledge (Hiebert & Lefevre, 1986).

A second contribution from cognitive science was the notion of a *buggy procedure*, a concept all-too-familiar to computer programmers. A piece of program code can sometimes execute without crashing and without producing error messages even though it contains some flaw or error that causes it to produce undesirable behaviors or unintended results. The buggy procedure concept moved front and center because empirical studies suggested that students' errors were not random. Instead, they appeared to fall into types. This finding ruled out the hypotheses that the errors were due

to failure to pay attention or lack of motivation, because those hypotheses implied random responding or failure to respond. Instead, the empirical data indicated that the majority of the errors were errors of commission. That is, the incorrect answers came about, not by the careless or incomplete execution of correct arithmetic procedures, but by the faithful execution of incomplete or misunderstood ("buggy") procedures (Brown & Burton, 1978; Sleeman, 1984).[2]

This observation was not entirely new. Teachers and educators already knew that there were patterns in students' erroneous answers, but cognitive scientists pushed the analysis further by *modeling* the buggy procedures that generated those answers. That is, they wrote, in some computer language, incorrect mathematical procedures, which, when executed, produced the same erroneous answers as children. The pioneers in this enterprise included John Seely Brown, Richard Burton, and Kurt VanLehn in the U.S. with respect to subtraction errors (Brown & Burton, 1978), and Derek Sleeman in the UK with respect to high-school algebra (Sleeman, 1984; Sleeman, Kelly, Martinak, Ward, & Moore, 1989). They published the first so-called *bug libraries*, repertoires of cognitive models that deviated from the correct mathematical skills in such a way as to generate the erroneous answers observed empirically (and no others; Brown & VanLehn, 1980). Given the paradigmatic cases of subtraction and algebra, cognitive scientists soon found buggy procedures up and down the mathematics curriculum, and in other subject matter areas as well (e.g., Spohrer, Soloway, & Pope, 1985). The occurrence of stable but senseless procedural errors was, and remains, a robust empirical phenomenon.

The buggy model construct suggested that instruction could be improved by finegrained tailoring of instruction to particular bugs.[3] Consider the so-called SMALLER-FROM-LARGER error, that is, to subtract the smaller digit in a column from the greater one, regardless of which is the minuend and which is the subtrahend, that is, which is "on top" or "on the bottom" of a column. Contrast this with another frequently observed subtraction bug, BORROW-NO-DECREMENT, in which the student 'borrows' into a column without decrementing the column 'borrowed' from, an error that violates the meaning of 'borrowing' (regrouping) in the context of arithmetic. Both errors are common in children's arithmetic (VanLehn, 1990). A student suffering the former bug might benefit from being asked, for example, *which number are you subtracting from which number*?, while a student suffering from the latter might benefit more from being asked, *when you borrow from a number*, *what happens to it*? A natural approach to building an intelligent tutoring system was therefore to turn the buggy models into diagnostic devices that identify exactly which bug, or combination of bugs, is responsible for a student's wrong answers, and then deliver tutoring messages designed to help the student correct it.

[2] This conclusion was undermined by later studies that indicated that bugs are not always stable within a student population, and that bug libraries do not always transfer between student populations (Tatsuoka, Birenbaum, & Arnold, 1989; Hennessy, 1994; Payne & Squibb, 1990). In yet another swing of the pendulum, Ben-Zeev and Ronald (2002) have recently shown that the stability of bugs depends on the level of abstraction at which they are described.

[3] See Sleeman and Brown (1982) for a collection of early papers that developed on this idea, and Wenger (1987) for a book-length review. Many of the models reported in these publications were explicit about the bugs, but stopped short of implementing tutoring components.

The implementation of this approach to building intelligent tutoring systems encountered multiple problems. Two of those problems directly shaped the development of CBM.

Two Gaps in the Research Literature

To maximize the pedagogical power of a tutoring message (or some other type of verbal instructional message), its content, formulation, and time of delivery need to be based on a theory of the relevant learning processes (Ohlsson, 1986, 1991). Such a theory should address two issues. First, what cognitive mechanism or mechanisms is supposed to translate the content of a verbal tutoring message into the appropriate change in the learner's current representation of the target skill? Second, to address educators' concerns about learning with understanding, the relevant learning theory should explain the role and function conceptual knowledge in the construction of new cognitive skills. The research literature of the 1980's disappointed on both accounts.

A Gap Between Learning and Verbal Instruction

By what cognitive mechanisms are buggy skills transformed into correct skills? In particular, by what cognitive mechanisms are verbal tutoring messages translated into improvements in a not-yet-mastered skill? In the early 1980s this was a timely question, because cognitive scientists had just began exploring computational models of learning. Anzai and Simon (1979) published the first computer simulation of skill acquisition through practice. Their model learned to perform the Tower of Hanoi puzzle in the course of four practice trials. It consisted of a problem solving component plus a set of *learning mechanisms*, cognitive processes that take a skill plus some additional information as inputs, and deliver a revised (improved) version of that skill as output. The particular learning mechanisms included in the Anzai and Simon model were not sufficient to explain the acquisition of a wide variety of cognitive skills, nor was it capable of taking instruction as input. It learned solely from its own experience in attempting the target task. But the model was important because it demonstrated that computer simulation of cognition had evolved to the point where it was possible to simulate changes in behavior over time, and not merely steady state performance. This paradigmatic achievement triggered an unprecedented flowering of the theoretical imagination: In the two decades that followed, more mechanisms for skill acquisition were proposed than during the prior history of cognitive research (Ohlsson, 2008).

Some of the processes programmed into the first computational models of skill acquisition were old favorites that were studied by scholars long before cognitive science, but they were transformed by being implemented as running computational mechanisms. For example, generalization (a.k.a. induction) had been discussed since Aristotle, but took a new form when applied to rules rather than categories or principles. Such a mechanism was incorporated into an early version of the ACT model of cognition (Anderson, 1982, 1983). Another mechanism with a long history is analogy, advocated, by among others, the American mathematician Polya (1945/1957, pp. 37–46), before being given a computational form by, among others, Falkenhainer, Forbus, and Gentner (1989). Some modelers focused on subgoaling, that

is, the process of breaking down a task into subtasks that could be mastered one at a time, and, once again, its computational incarnation (Rosenbloom & Newell, 1986) was rather different than the subgoaling concept that had been introduced into the study of problem solving by Duncker (1935/1974). Other mechanisms with a shorter pedigree were proposed as well. The effects of automaticity were modeled through processes that contracted unvarying sequences of steps into single steps (Neves & Anderson, 1981). Skills were also hypothesized to improve via the elimination of redundant or unnecessary steps (Neches, 1987; Ruiz & Newell, 1993; Shrager & Siegler, 1998). All implemented cognitive skill mechanisms described in the literature up to 2008 were reviewed in Ohlsson (2008).

These modes of learning shared two characteristics that are especially relevant for the present purposes. First, they emphasize learning from experience. For example, one type of model began solving an unfamiliar type of problem by using heuristic search or some other weak method, and then compiled information about the problem space discovered in the course of searching into more selective, task-specific heuristics (e.g., Langley, 1983, 1985). Another influential type of model attacked unfamiliar problems by searching for an analogy with some already mastered task (e.g., Falkenhainer et al., 1989). These and other types of experiential learning models did not take verbal instructional messages as inputs, and hence did not explain how such a message might engender an improvement in a not-yet-mastered cognitive skill. That is, they did not explain how tutoring works (nor were any claims to that effect made by the researchers proposing them).

A second shared feature of the first wave of cognitive learning mechanisms is that they capitalized on positive outcomes: correct answers, productive analogies, satisfied subgoals, useful shortcuts, and so on. What, after all, would be the point of generalizing an *erroneous* step? If a skill, or a component of a skill, generates incorrect steps, then nothing good can come of making that skill or skill component apply more broadly! Likewise with analogy: There is little to be gained by basing a new skill on an analogy with a previously learned, *incorrect* skill. Subgoals are only helpful if they truly slice the relevant task into independent and easily achievable subtasks; unreachable subgoals are unlikely to help. With the exception of learning by discrimination (Langley, 1983, 1985, 1987), the learning mechanisms that were explored in computational models of skill acquisition were mechanisms that cache successful steps for future use.

The focus on experiential learning from positive outcomes made cognitive models of learning and intelligent tutoring systems awkward dance partners, the frequent claims that tutoring systems were 'based on' this or that cognitive model notwithstanding. As long as the models did not take verbal tutoring messages as inputs, they had few implications for how tutoring messages should be written, and for when they should be delivered. In addition, teaching to errors is a natural way to tutor: After all, why interrupt the student and instruct when he or she is on the right track? It makes more intuitive sense to step in when the student shows sign of needing help. But theories and models that focus on the use of positive outcomes have few implications for how students can learn from their errors.

In short, the ambition to base the design of intelligent tutoring systems on theories of learning encountered a gap in the research literature: Most computational theories of learning proposed in the 1980s were built around cognitive mechanisms for experiential learning that primarily capitalize on positive outcomes, while what was needed to base

tutoring technology on a theoretical basis were hypotheses about how verbal tutoring messages about errors and other negative outcomes can be utilized by a student to improve a not-yet-mastered skill. To overcome this mismatch between mode of learning and mode of tutoring, the field needed a theory of learning from tutoring messages about errors.

A Gap Between Declarative and Procedural Knowledge

Although mathematics educators worried about the lack of conceptual understanding on the part of students, the educational research literature of the 1980s did not provide precise statements and hypotheses about what it means to understand a mathematical procedure or skill. There were no attempts by educational researchers to formalize conceptual knowledge, or to pose precise hypotheses about how it functions and by which cognitive processes conceptual knowledge might guide the acquisition of new mathematical skills. There was no rigorous educational theory about how understanding and performance interact, only a strong sense that they ought to interact (Hiebert & Lefevre, 1986). In particular, there was no theory of how understanding (declarative knowledge) guides the acquisition of a skill (procedural knowledge).

Within cognitive science, the dominant model of declarative knowledge was inherited from formal logic: Declarative knowledge consists of propositions. The latter serve as premises for inference rules that warrant the derivation of new propositions. If the inference rules are valid and the premises are true, then the conclusion is true as well. In this model, reasoning starts with declarative knowledge, the premises, and ends with new declarative knowledge, the conclusion. The process never breaks out of the declarative realm. It is all about truth, not action or goal attainment. However, the educators' concept of learning and performing with understanding required a bridge from thought to action such that declarative knowledge guides the acquisition and execution of procedural knowledge.

There were snippets of relevant theory spread out here and there across the cognitive landscape like flowers in a meadow, each pretty in its own way but very different from each other. Philosophers had developed *practical logic*, a theory of inferences that end with an action instead of conclusion, but they applied it primarily to questions about ethics. Simon (1972) published a brief semi-formal calculus of the concept "can", as in "can do X." Hayes-Roth, Klahr, and Mostow (1981) and Mostow (1983), building on a theoretical analysis of John McCarthy, created an Artificial Intelligence system that could translate a high-level piece of advice into a specific action. The most extensively analyzed example was the transformation of the advice *avoid taking points* into the action *play a low card* in the context of playing bridge. The transformation required a hundred special-purpose inference rules. Neves and Anderson (1981) analyzed an example of how a geometry theorem can be transformed into a proof strategy by gradual and piecemeal transformation into production rules. This transformation was achieved by a set of special-purpose interpretative production rules.[4] Greeno, Riley, and Gelman (1984) and Smith, Greeno, and Vitolo (1989) described a computational model of children's transfer of counting skills to a non-standard counting task. Their proposed process looked very much like means-ends analysis, and the children's hypothesized

[4] See Anderson (1982), Table 2, p. 376, and Anderson (1983), Table 6.1, pp. 222–223.

conceptual knowledge looked more like problem solving operators than like concepts or principles.

None of these efforts, interesting though they were, seemed a satisfactory account of how the mind translates thought into action. The cognitive mechanisms described by Hayes-Roth, Klahr, and Mostow, Neves and Anderson, and Greeno and co-workers were complicated and in some cases required hundreds of cognitive steps. It seemed doubtful that school children who understand mathematics undergo some process that is even remotely similar to the processes envisioned in these models, and even less plausible that prompting the envisioned processes would help unsuccessful children acquire mathematical skills with understanding. I began to think that the very idea of deriving procedural knowledge from declarative knowledge was fundamentally flawed. As a theoretical exercise, I derived the standard multi-column routine for subtraction with regrouping from its algebraic premises. The derivation turned out to require twenty-five pages of formulas.[5] This exercise settled my mind that this was a dead-end path. The interaction between declarative and procedural knowledge cannot be conceived as a derivation of the correct performance from its conceptual rationale. The relation between thought and action in mathematics and elsewhere had to be understood differently.

To summarize, the goal to base the detailed design of tutoring systems on learning theory posed two theoretical problems. First, to the extent that a tutoring system is programmed to intervene and teach when students need help, the relevant learning theory should explain how, by which cognitive processes, students learn from their errors. Second, because the desired pedagogical goal is that students should understand what they learn, the relevant learning theory should specify the role and function of declarative knowledge in the acquisition of cognitive skills. The research literature of the 1980s provided no solution to either problem. The invention of CBM was a response to this situation.

Two Gaps, One Bridge

The exact moment of invention is lost in the fog of memory, but it produced a novel learning mechanism that explained both how a person can learn from his or her errors and how declarative knowledge guides that process. The implications of this theory for tutoring became the CBM approach to ITS design. The following rational reconstruction breaks the development process down into three successive steps.

Step 1: From Propositions to Constraints

The first step was to go beyond the ancient idea that declarative knowledge consists of propositions. The key insight was that discourse about errors and error correction is normative in character. During skill practice, the question for the learner is which action he or she *ought to* take next, and what the solution to his or her current practice problem *ought to* look like.

[5] The derivation is available in an unpublished technical report that does not exist electronically. I will gladly mail it to anyone who is interested in owning a copy of this cognitive curiosity.

This observation suggested that the function of declarative knowledge is not to enable a learner to *derive* the action to be taken next, but to *evaluate* the outcomes of tentative actions, that is, actions taken in the absence of sufficient reasons. From this point of view, the units of declarative knowledge are best thought of as *constraint*s, knowledge elements that encode prescriptive rather than descriptive knowledge. The type of constraint used in CBM has the general form, "when such-and-such conditions are the case, then such-and-such other conditions ought to be the case as well" (or else something has gone wrong). For example, *when driving a car in New Zealand*, *the driver had better be driving on the left side of the road* (or else he or she violates the traffic laws of that country). Clearly, a speed limit is not a description of actual behavior, but a prescription. Formally, constraints of this sort take the form of ordered pairs of patterns, <Cr, Cs>, where each pattern is a conjunction of conditions. Cr is a *relevance criterion* that circumscribes the set of situations for which the constraint applies (*when driving in New Zealand*), and Cs is a *satisfaction criterion* that determines whether the constraint is satisfied (*drive on the left side of the road*). The set of constraints that apply to a problem type or in a particular task environment is called a *constraint base*.

Given a constraint-base, the detection and identification of errors can be reduced to pattern matching: Match the relevance conditions of all available constraints against the current situation or problem state; for each relevant constraint, match the satisfaction patterns as well. If the relevance condition does not match, or if both conditions are satisfied, there is no evidence of error. If a relevance condition is satisfied but the associated satisfaction condition is not, then some error was made on the way to the current problem state. The content of the violated constraint (or constraints) specifies the nature of the error. Examples of constraints and learning events of this sort are available in the original papers (Ohlsson, 1993, 1996a, 1996b, Ohlsson & Rees, 1991a, 1991b).

Step 2: From Error Detection to Error Correction

The next question was how errors, once detected, can be corrected, and how the correction can be guided by the content of the constraints. Given the constraint formalism, the computational problem to be solved can be stated as follows: Given (a) a learning scenario in which the learner is practicing a not-yet-mastered cognitive skill; (b) the learner's current mental representation of that skill; (c) a particular situation or problem state; (d) an (incorrect) action performed in that situation; and (e) an undesirable outcome, that is, a violation of some constraint, how can the learner's cognitive system compute the warranted revision of his or her current representation of the target skill?

Working with Ernest ("Ted") Rees, a brilliant programmer at LRDC in Pittsburgh, I invented a learning mechanism that we came to call *constraint-based rule specialization*. This learning mechanism is based on the idea of constraining a rule-being-learned to situations in which it does not generate errors. The binary structure of constraints affords two revisions of an error-producing rule. First, a rule can be specialized so as to not apply in situations in which the constraint is relevant. If the constraint is not relevant, it cannot be violated. Glossing over a mountain of technical details, this type of specialization can be accomplished by adding the *negation* of the relevance

condition of the violated constraint, Cr, to the applicability condition for the skill element. A second revision of the erroneous rule is to specialize the rule to situations in which the constraint is guaranteed to be satisfied. Once again glossing over the technicalities, this can be done by adding the satisfaction condition to the applicability condition of the relevant rule. Together, these two revisions produce two new rules, each more specific than the original, erroneous rule. Both are added to the learner's rule base. The old, overly general rule remains in the rule base, but with lower priority than its more specific descendants. The augmented rule set will not produce the type of error that triggered the revision. The reader is referred to the relevant publications for formal details and multiple examples (Ohlsson, 1993, 1996a, 1996b; Ohlsson & Rees, 1991a, 1991b).

We implemented constraint-based rule specialization within a home grown rule based architecture called HS, and verified that it affords successful learning from errors in multiple task domains, including children's counting, subtraction with regrouping, and college chemistry. We also verified that it generates negatively accelerated learning curves like those observed in human learning (Ohlsson, 1993, 1996b; Ohlsson & Jewett, 1997). In a later application, Choi and Ohlsson (2011) re-implemented this mechanism in *Icarus*, a cognitive architecture developed by Patrick Langley and co-workers (Langley & Choi, 2006). Icarus is not a rule-based system, but the learning mechanism worked well in multiple application domains that included landscape navigation and the Blocks World domain frequently used in Artificial Intelligence research. The constraint-based specialization is a general, versatile learning mechanism.

The most novel feature of constraint-based specialization is that it claims that skills start out overly general, and are gradually specialized to a particular task environment by incorporating the knowledge encoded in the constraints (the declarative knowledge) into the rules (the procedural knowledge). This principle explains the correction of both errors of omission and errors of commission. This seemed then, and seems to me still, a plausible view of how conceptual knowledge guides action during skill practice. The elements of the skill-to-be-learned are operating within the context of the universe of solution paths circumscribed by the relevant constraint base.

Step 3: From Learning to Tutoring

If constraint-based specialization corresponds at least approximately to the computations performed by a learner's brain when confronted with a negative outcome, how can a tutor (human or machine) facilitate learning? What is the role and function of the tutor in this scenario? If the tutor has a larger constraint base than the student, then the tutor can create learning opportunities by catching constraint violations that the student does not heed because he or she does not know the relevant constraint. The tutor can signal the constraint violation and instruct the learner by communicating the relevant constraint and highlight how it was violated by the action the learner took. The constraint-based specialization process in the learner's head can then revise his or her current representation of the target skill in the way described in the previous section.

We simulated this scenario in collaboration with Andreas Ernst, a graduate student from Freiburg University in Germany who visited LRDC for a year. We tutored our HS

model in subtraction with regrouping. The model was given basic capabilities such as single-digit subtraction, moving attention from column to column, and writing a digit in a column. It was then presented with multi-column subtraction problems but no procedural knowledge of how to solve them. The model initially made nonsensical errors, just like human students. For each error, we interrupted the model and typed in a 'tutoring message' in the form of a constraint, and triggered the constraint-based learning mechanism. Eventually, the model performed subtraction correctly. Depending on which constraints we gave it, the model would learn either the regrouping or the augmentation algorithms for multi-column subtraction(Ohlsson, Ernst, & Rees, 1992).

The obvious next step was to let the student and the machine switch roles. The implications for ITS design were, in principle, obvious (Ohlsson, 1992): A constraint-based ITS was to be built around a set of constraints that define the correct subject matter for a particular domain or subject matter topic. Such an ITS would apply the constraints to each new problem state and flag violated constraints. Pre-formulated instructional messages would be associated with the constraints, and presented to the student when one or more constraints are violated. This is the core of the constraint-based approach. One notable advantage is that the constraint-based approach does not require empirical studies of students' errors or the compilation of bug libraries, because constraints encode correct domain knowledge. This seemed to me then, and seems to me still, a simpler and more elegant design for an ITS than to organize it round either a bug library or an expert model of the target skill.

Nevertheless, the'92 paper that proposed the constraint-based approach almost did not happen. It was an afterthought, written with the expectation that it would round out the multi-year research program on learning from error. That line of research had reached a natural end point. It was not obvious where to take the work next, and the *Zeitgeist* in the field of cognition and education had moved away from procedural skills by the end of the 1980s. Being able to do multi-column arithmetic with paper and pencil appeared less and less important in the age of calculators and computers. Cognitive researchers executed a pivot from a concern with the understanding of procedural skills in mathematics to a focus on conceptual change in science learning.

The constraint-based approach might thus have languished as an interesting but unproven idea. However, geopolitics and personal history intervened. The'92 concept paper was read by someone who knew what to do with the ideas expressed in it. In the spring of 1998, Antonija ("Tanja") Mitrović came by my office at the University of Illinois at Chicago. She was in the process of re-locating herself and her family from the disintegrating country of Yugoslavia to their new home in New Zealand. We discussed the CBM approach, how it might be used to build a tutoring system to teach the skills involved in using a common database query language, SQL, and how one might go about empirically verifying its pedagogical effectiveness. The story of the first CBM tutoring system, SQL-Tutor, is her story; see the companion paper in this issue. Her research group has since designed, implemented, deployed, and evaluated multiple CBM systems. Their hard work has proven beyond doubt that CBM is a practical, elegant, and versatile design theory for intelligent tutoring (Mitrović, 2003, 2006, 2012; Mitrović, Martin, & Mayo, 2002; Mitrović, Martin, & Suraweera, 2007; Mitrović & Ohlsson, 1999, 2006; Mitrović, Suraweera, Martin, & Weerasinghe, 2004; Suraweera & Mitrović, 2004; Weerasinghe & Mitrović, 2005).

Beyond CBM

Looking back, the rationale for the constraint-based approach still appears valid. People do learn from their errors, and the specialization of the applicability conditions of skill elements is still a plausible hypothesis about the underlying cognitive process. I know of no subsequent research that calls this hypothesis into question. Similarly, the CBM design for ITS has proven its value over and over again in the success of the constraint-based systems coming out of various research grouips, primarily Mitrović's group at Canterbury University, but also our team in Chicago (Fossati et al., 2009) as well as others (Menzel, 2006; Roll, Aleven, & Koedinger, 2010). The experience of implementing and deploying these systems has not revealed fundamental flaws in the approach that limit its application or usefulness.

There are, of course, open issues that might benefit from future research, including how the system developer can decide whether a constraint base is complete and how complete a constraint base has to be to support effective tutoring. Another open issue is how a constraint-based ITS should select practice problems for maximal learning gains. Such issues are probably best resolved in the practice of implementing CBM systems for particular task domains, as opposed to deriving detailed prescriptions in a top down manner.

An open question of deep theoretical interest is where the constraints come from. The theory of learning from error explains how they function, not how they are acquired. In some cases, they originate in the task environment. For example, in the HS chemistry model (Ohlsson, 1993, 1996b), the constraints encoded the principles of the co-valent bond, which are explicitly stated in any college chemistry text. In the counting application (Ohlsson, 2006; Ohlsson & Rees, 1991a), we relied on the hypothesis, advanced by developmental psychologists, that the constraints that define one-to-one mapping are innate, or at least emerge in the first few months of life (Gelman & Meck, 1983). In puzzle tasks like Tower of Hanoi and the Nine-Dot Problem (Choi & Ohlsson, 2011), the constraints are built into the task instructions (*do not put a larger disc on a smaller one*; *only use four lines*). In other cases, the origin of the relevant constraints is obscure. This does not affect the role and function of constraints in tutoring: The point of CBM is that the tutoring system has a larger constraint-base than the learner, and hence can create learning opportunities by identifying constraint violations that the learner overlooks. A hypothesis about the origin of constraints would be theoretically interesting but it may or may not have implications for the implementation of constraint-based tutoring systems.

Another broad issue with greater potential impact on the implementation of tutoring is that skills are acquired in other ways than through the correction of errors. With hindsight, it is obvious that the excitement generated by buggy models led myself and perhaps others as well to overemphasize error correction as the main mode of learning during skill practice, and hence to regard the teaching to errors as the main mode of tutoring. But empirical studies of human tutors have since shown that error correction is only one type of tutoring move (Cade, Copeland, Person, & D'Mello, 2008; Graesser, Person, & Magliana, 1995; Lu, Eugenio, Kershaw, Ohlsson, & Corrigan-Halpern, 2007; Ohlsson et al., 2007; Person, Graesser, Kreutz, & Pomeroy, 2003), indicating that tutors intuitively recognize the existence of multiple modes of learning.

How many modes of learning might there be? In 2008, I reviewed all published cognitive simulation models of skill acquisition (Ohlsson, 2008). It turns out that there are fewer learning mechanisms than there are models, so if we slice the models into their component mechanisms and sort the latter by similarity, we are left with a small repertoire of distinct modes of learning. These modes claim that people learn (a) by proceduralizing direct verbal instructions (*when the green light comes on*, *press the XYZ button*); (b) via practical reasoning from declarative knowledge; (c) through transfer from previously mastered skills; (d) by observing a demonstration or studying a worked example; (e) by decomposing the top goal of the target skill into subgoals; (f) by caching positive outcomes for future use; (g) by detecting and correcting errors; (h) by eliminating redundant and unnecessary steps; and (i) by accumulating implicit knowledge of the statistical regularities of the task environment and using those regularities to optimize the target skill. In short, the repertoire of skill acquisition models implemented to date implicitly claims that there are nine distinct modes of cognitive change during skill practice.

Each learning mechanism, each mode of learning, implies a distinct mode of instruction. Consider learning from positive feedback. What capabilities are required to support this mode of learning? The paradox is that in order to receive positive feedback, the learner has to do the right thing. But if he or she knows enough to do the right thing, what is there for him or her to learn? The answer is presumably that learners take tentative steps during skill practice. The learner does not know with certainty that the step just taken was the right one. Positive feedback (*yes*, *that's right*; *go on*) reduces the uncertainty by confirming that the tentative step was indeed correct. However, if this type of feedback is delivered every time the learner does something correctly, the output from the tutor would be tedious and repetitive, and the learner would soon stop paying attention. The main issue in implementing this mode of instruction is to identify the exact conditions under which positive feedback is likely to be helpful. Mitrović, Ohlsson, and Barrow (2013) suggested a set of such conditions.

As a second example, consider learning through transfer from a previously acquired skill, or from a previously solved practice problem. What are the capabilities a tutoring system would need to teach to this learning mode? First, the system has to be able to determine which practice problems in its problem base are analogous to the problem the learner is working on. This can be done most simply by creating a pre-determined set of practice problems (which is the case in many, perhaps most, ITSs) and pre-storing transfer links among them. If an ITS generates practice problems dynamically, then the issue of how to establish or predict transfer from previously solved practice problems becomes more complicated. The ITS might need to implement something akin to the Structure Mapping Engine developed by Falkenhainer et al. (1989). Second, an ITS that is to tutor by encouraging transfer would need some metric for assessing the probability of transfer or the amount of transfer to be expected. One again, this assessment can be pre-computed if the practice problems are pre-determined. Finally, the ITS would need some criteria for identifying situations in which appeal to prior solutions is likely to be more productive than alternative tutoring modes. In such situations, the ITS might say something like, *Can you think of a similar problem*? (Polya, 1945/1957, p. 98). A key issue is how to assess what the student does in response. If the student recalls a promising transfer problem but nevertheless remains stuck, what is the appropriate tutor response?

As a final example, consider optimization in the asymptotic part of the learning curve (Neches, 1987; Ruiz & Newel, 1993; Shrager & Siegler, 1998; Siegler & Araya, 2005). How can an ITS support this mode of learning? If there exists a shortcut, an ITS could simply show it to the learner (*That's correct but there is a quicker way to get the same result…*). One possible way to show a shortcut is to present an example problem that is solved twice, once in the way the student has mastered and once with the shortcut. If possible, both solution paths could be displayed side by side on the same screen. After the presentation, the learner should be presented with practice problems in which the shortcut applies. The existence of short cuts in problems of a particular type could be predetermined or computed dynamically by a problem solving module.

All nine modes of learning identified in Ohlsson (2008) can be supported by instruction, but as the three examples above illustrate, the computational capabilities needed are unique for each mode. Implementing an ITS that can teach to all nine modes of learning is therefore considerably more labor intensive than building systems that only teach to one or a small handful of learning modes. But it is plausible that if a tutoring system could teach to all nine modes, it would increase the students' learning gains. If a tutoring system only supports learning in modes A, B, and C, then any situation that affords learning through mode D is a missed learning opportunity. Adding modes of instruction increases the proportion of learning opportunities that the learner can benefit from. Empirical studies of tutoring dialogues support the idea that human tutors engage in a wide variety of tutoring moves (Graesser et al., 1995; Lu et al., 2002; Ohlsson et al., 2007; Person et al., 2003). In a few cases, there is direct empirical support for the idea that the pedagogical power of a tutoring system increases when the system is augmented to teach to additional modes of learning. Mitrović et al. (2013) added the capability of teaching through positive feedback to SQL-Tutor, and saw the time to mastery cut in half. Similarly, Salden, Aleven, Schwonke, and Renkl (2010) and Schwonke et al. (2009) have demonstrated that a tutoring system becomes more effective when it is extended with the capability to teach through worked examples. The natural extrapolation of such findings is that to be maximally effective, a tutoring system needs to teach to all nine modes of learning.

Twenty-five years ago, the constraint-based specialization theory of learning from error led to CBM, a novel and useful approach to ITS design. The fact that CBM derived from a theory of learning suggests that a better learning theory might generate an even better tutoring technology (Ohlsson, 1991). In the next twenty-five years, the Nine Modes Theory of learning might lead to a Multiple Modes Theory of Instruction (MMTI), a design for ITSs that are maximally effective because they teach to all modes of learning and hence can help the student capitalize on every learning opportunity. Perhaps we are still in the early stages of the development of a tutoring technology that enables tutoring systems to outperform human tutors.

References

Anderson, J. R. (1976). *Language, memory, and thought*. Hillsdale, NJ: Erlbaum.
Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, *89*, 369–406.
Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.
Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review*, *86*, 124–140.

Ben-Zeev, T., & Ronald, J. (2002). Is procedure acquisition as unstable as it seems? *Contemporary Educational Psychology*, *27*, 529–550.

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

Brown, J. S., & VanLehn, K. (1980). Repair theory: a generative theory of bugs in procedural skills. *Cognitive Science*, *4*, 379–426.

Brownell, W. A. (1947). The place of meaning in the teaching of arithmetic. *Elementary School Journal*, *47*, 256–265.

Cade, W. L., Copeland, J. L., Person, N. K., & D'Mello, S. K. (2008). Dialogue modes in expert tutoring. In B. P. Woolf, E. Aïmeur, R. Nkambou, & S. Lajoie (Eds.), *Intelligent tutoring systems* (pp. 470–479). New York: Springer-Verlag.

Carver, S. M., & Klahr, D. (Eds.) (2001). *Cognition and instruction: twenty-five years of progress*. Mahwah, NJ: Erlbaum.

Chi, M., & Ohlsson, S. (2005). Complex declarative learning. In K. J. Holyoak, & R. G. Morrison (Eds.), *The Cambridge handbook of thinking and reasoning* (pp. 371–399). Cambridge, MA: Cambridge University Press.

Choi, D., & Ohlsson, S. (2011). Effects of multiple learning mechanisms in a cognitive architecture. In L. Carlson, C. Hölscher, & T. Shipley (Eds.), *Proceedings of the 33rd annual meeting of the cognitive science society* (pp. 3003–3008). Austin, TX: Cognitive Science Society Boston.

Dillon, R. F., & Sternberg, R. J. (Eds.) (1986). *Cognition and instruction*. New York: Academic Press.

Duncker, K. (1935/1974). *Zur Psychologie des produktiven Denkens* (Dritter Neudruck). Berlin, Germany: Springer Verlag.[English version: Duncker, K. (1945). On problem-solving. Psychological Monographs, vol. 58, Whole No. 270.]

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: algorithm and examples. *Artificial Intelligence*, *41*, 1–63.

Fossati, D., Di Eugenio, B., Brown, C. W., Ohlsson, S., Cosejo, D. G., & Chen, L. (2009). Supporting computer science curriculum: exploring and learning linked lists with iList. *IEEE Transactions on Learning Technologies*, *2*, 107–120.

Gelman, R., & Meck, E. (1983). Preschoolers' counting: principles before skill. *Cognition*, *13*, 343–359.

Graesser, A. C., Person, N., & Magliana, J. P. (1995). Collaborative dialogue patterns in naturalistic one-on-one tutoring. *Applied Cognitive Psychology*, *9*, 359–387.

Greeno, J., Riley, M. S., & Gelman, R. (1984). Conceptual competence and children's counting. *Cognitive Psychology*, *16*, 94–143.

Hayes-Roth, F., Klahr, P., & Mostow, D. (1981). Advice taking and knowledge refinement: an iterative view of skill acquisition. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 231–253). Hillsdale, NJ: Erlbaum.

Hennessy, S. (1994). The stability of children's mathematical behavior: when is a bug really a bug? *Learning and Instruction*, *3*, 315–338.

Hiebert, J. (Ed.) (1986). *Conceptual and procedural knowledge: The case of mathematics*. Hillsdale, NJ: Erlbaum.

Hiebert, J., & Lefevre, P. (1986). Conceptual and procedural knowledge in mathematics: an introductory analysis. In J. Hiebert (Ed.), *Conceptual and procedural knowledge: The case of mathematics* (pp. 1–27). Hillsdale, NJ: Erlbaum.

Klahr, D. (Ed.) (1976). *Cognition and instruction*. Hillsdale, NJ: Erlbaum.

Langley, P. (1983). Learning search strategies through discrimination. *International Journal of Man-Machine Studies*, *18*, 513–541.

Langley, P. (1985). Learning to search: from weak methods to domain-specific heuristics. *Cognitive Science*, *9*, 217–260.

Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *(1987). production system models of learning and development* (pp. 99–161). Cambridge, MA: MIT Press.

Langley, P., & Choi, D. (2006). Learning recursive control programs from problem solving. *Journal of Machine Learning Research*, *7*, 493–518.

Lesgold, A. M., Pellegrino, J. W., Fokkema, S. D., & Glaser, R. (Eds.) (1978). *Cognitive psychology and instruction*. New York: Plenum Press.

Lu, X., Eugenio, D., Kershaw, T., Ohlsson, S., & Corrigan-Halpern, A. (2007). Expert vs. non-expert tutoring: dialogue moves, interaction patterns and multi-utterance turns. *Lecture Notes in Computer Science*, *4394*, 456–467.

Menzel, W. (2006). Constraint-based modeling and ambiguity. *International Journal of Artificial Intelligence in Education*, *16*, 29–63.

Mitrović, A. (2003). An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education*, *13*, 173–197.

Mitrović, A. (2006). Large-scale deployment of three intelligent web-based database tutors. *Journal of Computing and Information Technology*, *14*, 275–281.

Mitrović, A. (2012). Fifteen years of constraint-based tutors: what we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, *22*(1–2), 39–75.

Mitrović, A., Martin, B., & Mayo, M. (2002). Using evaluation to shape ITS design: results and experiences with SQL-tutor. *International Journal of User Modeling and User-Adapted Interaction*, *12*, 243–279.

Mitrović, A., Martin, B., & Suraweera, P. (2007). Intelligent tutors for all: constraint-based modeling methodology, systems and authoring. *IEEE Intelligent Systems*, *22*, 38–45.

Mitrović, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a data-base language. *International Journal of Artificial Intelligence and Education*, *10*, 238–256.

Mitrović, A., & Ohlsson, S. (2006). Constraint-based knowledge representation for individualized instruction. *Computer Science and Information Systems*, *3*, 1–22.

Mitrović, A., Ohlsson, S., & Barrow, D. K. (2013). The effect of positive feesdback in a constraint-based intelligent tutoring system. *Computers & Education*, *60*, 264–272.

Mitrović, A., Suraweera, P., Martin, B., & Weerasinghe (2004). DB-suite: experiences with three intelligent web-based database tutors. *Journal of Interactive Learning Research*, *15*, 409–432.

Mostow, D. J. (1983). Machine transformation of advice into a heuristic search procedure. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 367–404). Palo Alto, CA: Tioga.

Neches, R. (1987). Learning through incremental refinement of procedures. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development* (pp. 163–219). Cambridge: MIT Press.

Neves, D. M., & Anderson, J. R. (1981). Knowledge compilation: mechanisms for the automatization of cognitive skills. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 57–84). Hillsdale: Erlbaum.

Ohlsson, S. (1986). Some principles of intelligent tutoring. *Instructional Science*, *14*, 293–326.

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

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

Ohlsson, S. (1993). The interaction between knowledge and practice in the acquisition of cognitive skills. In S. Chipman, & A. L. Meyrowitz (Eds.), *Foundations of knowledge acquisition: cognitive models of complex learning* (pp. 147–208). Boston: Kluwer.

Ohlsson, S. (1994). Declarative and Procedural knowledge. In T. Husen, & T. Neville-Postlethwaite (Eds.), *The International Encyclopedia of Education* (vol. 3, 2nd ed., pp. 1432–1434). London: Pergamon Press.

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

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

Ohlsson, S. (2006). Order effects in constraint-based skill acquisition. In F. E. Ritter, J. Nerb, T. O'Shea, & E. Lehtinen (Eds.), *In order to learn: How ordering effects in machine learning illuminates human learning and vice versa* (pp. 151–165). New York, NY: Oxford University Press.

Ohlsson, S. (2008). Computational models of skill acquisition. In R. Sun (Ed.), The Cambridge handbook of computational psychology (pp. 359–395). Cambridge.

Ohlsson, S., Di Eugenio, B., Chow, B., Fossati, D., Lu, X., & Kershaw, T. (2007). Beyond the code-and-count analysis of tutoring dialogues. In R. Luckin, K. Koedinger, & J. Greer (Eds.), *Artificial intelligence in education: building technology rich learning contexts that work* (pp. 349–356). Amsterdam, The Netherlands: IOS Press.

Ohlsson, S., Ernst, A. M., & Rees, E. (1992). The cognitive complexity of doing and learning arithmetic. *Journal of Research in Mathematics Education*, *23*, 441–467.

Ohlsson, S., & Jewett, J. J. (1997). Ideal adaptive agents and the learning curve. In J. Brzezinski, B. Krause, & T. Maruszewski (Eds.), *Idealization VIII: modeling in psychology* (pp. 139–176). Amsterdam, The Netherlands: Rodopi.

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

Ohlsson, S., & Rees, E. (1991b). Adaptive search through constraint violation. *Journal of Experimental and Theoretical Artificial Intelligence*, *3*, 33–42.

Payne, S. J., & Squibb, H. R. (1990). Algebra mal-rules and cognitive accounts of errors. *Cognitive Science*, *14*, 445–481.

Person, N. K., Graesser, A. C., Kreutz, R. J., & Pomeroy, V. (2003). Simulating human tutor dialog moves in AutoTutor. *International Journal of Artificial Intelligence in Education*, *12*, 23–39.

Polya, G. (1945/1957). *How to solve it* (2nd ed.) New York: Doubleday.

Roll, I., Aleven, V., Koedinger, K. R. (2010). The Invention lab: Using a Hybrid of Model Tracing and Constraint-Based Modeling to Offer Intelligent Support in Inquiry Environments. V. Aleven, J. Kay, & J. Mostow, (Eds.), Lecture Notes in Computer Science, No. 6094, Part I pp. 115–124. Berlin: Springer-Verlag.

Rosenbloom, P., & Newell, A. (1986). The chunking of goal hierarchies: a generalized model of practice. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (vol. 2, pp. 247–288). Los Altos, CA: Kaufmann.

Ruiz, D., & Newell, A. (1993). Tower-noticing triggers strategy-change in the tower of Hanoi: a Soar model. In P. S. Rosenbloom, J. E. Laird, & A. Newell (Eds.), *The soar papers: research on integrated intelligence* (vol. 2, pp. 934–941). Cambridge, MA: MIT Press.

Salden, R., Aleven, V., Schwonke, R., & Renkl, A. (2010). The expertise reversal effect and worked examples in tutored problem solving. *Instructional Science*, *38*, 289–307.

Schwonke, R., Renkl, A., Krieg, C., Wittwer, J., Aleven, V., & Salden, R. (2009). The worked-example effect: not an artefact of lousy control conditions. *Computers in Human Behavior*, *25*, 258–266.

Shrager, J., & Siegler, R. S. (1998). A model of children's strategy choices and strategy discoveries. *Psychological Science*, *9*, 405–410.

Siegler, R., & Araya, R. (2005). A computational model of conscious and unconscious strategy discovery. In R. V. Kail (Ed.), *Advances in child development and behavior* (vol. 33, pp. 1–42). Oxford, UK: Elsevier.

Silver, E. A. (1986). Using conceptual and procedural knowledge: a focus on relationships. In J. Hiebert (Ed.), *Conceptual and procedural knowledge: The case of mathematics* (pp. 181–198). Hillsdale, NJ: Erlbaum.

Simon, H. A. (1972). On reasoning about actions. In H. A. Simon, & L. Siklossy (Eds.), *Representation and meaning* (pp. 414–430). Englewood Cliffs, NJ: Prentice-Hall.

Sleeman, D. (1984). An attempt to understand students' understanding of basic algebra. *Cognitive Science*, *8*, 387–412.

Sleeman, D., & Brown, J. S. (Eds.) (1982). *Intelligent tutoring systems*. London, UK: Academic Press.

Sleeman, D., Kelly, A. E., Martinak, R., Ward, R. D., & Moore, J. L. (1989). Studies of diagnosis and remediation with high school algebra students. *Cognitive Science*, *13*, 551–568.

Smith, D. A., Greeno, J. G., & Vitolo, T. M. (1989). A model of competence for counting. *Cognitive Science*, *13*, 183–211.

Spohrer, J. C., Soloway, E., & Pope, E. (1985). A goal/plan analysis of buggy Pascal programs. *Human-Computer Interaction*, *1*, 163–205.

Suraweera, P., & Mitrović, A. (2004). An intelligent tutoring system for entity relationship modelling. *Artificial Intelligence in Education*, *14*, 375–417.

Tatsuoka, K. K., Birenbaum, M., & Arnold, J. (1989). On the stability of students' rules of operation for solving arithmetic problems. *Journal of Educational Measurement*, *26*, 351–361.

VanLehn, K. (1990). *Mind bugs: The origins of procedural misconceptions*. Cambridge, MA: MIT Press.

Weerasinghe, A., & Mitrović, A. (2005). Facilitating deep learning through self-explanation in an open-ended domain. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, *9*, 1–17.

Wenger, E. (1987). *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*. Los Altos, CA: Kaufmann.

Winograd, T. (1972). *Understanding natural language*. New York: Academic Press.

Winograd, T. (1975). Frame representations and the declarative/procedural controversy. In D. G. Bobrow, & A. Collins (Eds.), *Representation and understanding: studies in cognitive science* (pp. 185–210). New York: Academic Press.