In E. Scanlon & T. O'Shea (Eds.), New Directions in Educational Technology. (pp. 23-31) (1992) Berlin: Springer-Verlag.

Table 2 cont.

For any technology, ***institutionalization*** depends on:

Powerful advocate.
To the degree that there is a budget-controlling administrator who is a strong advocate, the more likely it is that institutionalization will occur.
There are some classroom management variables that affect both ***initial implementation by teachers*** and ***continued use by teachers*** of computer technology.
Activity-centered classrooms.
If teachers structure classrooms around activity centers, then it is easy to incorporate computers into classrooms by adding one or two computers to the activity centers. This style allows for effective use in low student-computer ratio settings.
Whole-class teaching.
If a teacher normally teaches to the whole class at one time, she has several options for trying to deal with the classroom management problem:

a) Some students miss the lesson.
If there are one or two computers in the classroom, the teacher may let a few students, who can afford to miss the lesson, work on computers at the same time as she conducts the lesson with the class. This can lead to problems about making up work. Teachers do not like to do this because they feel their lessons are important for everyone, and so this strategy works against continued use.

b) Works with whole class on computers together.
This is what happened in Columbus ACOT ("Apple Classroom of Tomorrow") classroom with 1-1 student-computer ratio (computers mostly sit idle). Normally this strategy is implemented by going to computer labs, which is somewhat disruptive of lesson continuity. This strategy works somewhat better than (a) for continued use.

c) Teacher uses computer for demonstrations.
If there is only one computer, then by using large screen projection, the teacher can run demonstrations on the computer. Effectiveness of this strategy depends on how much involvement the teacher can elicit from students.

d) Team turn-taking.
Tom Snyder's Search Series can be used where four teams take turns at a computer, and plan their next move while they wait for their next turn. This strategy is quite effective for continued use.

3

A Workbench for Discovering Task Specific Theories of Learning

Kurt VanLehn
Departments of Psychology and Computer Science
Carnegie-Mellon University, Pittsburgh, PA 15217 U.S.A.

Abstract: This chapter examines why learning theories expressed as artificial intelligence programs have not had much direct effect on education and training. It suggests a new research direction.

Keywords: ACM, ACT*, addition, arithmetic algorithms, Artificial Intelligence, CASCADE, CIRRUS, cognitive behaviour, cognitive science, generalization hierarchy, immediate feedback, impasse, knowledge representation, learning theories, machine learning, METADENDRAL, physics, psychology, SAPA, skill acquisition, SOAR, subtraction, task analysis, TETRAD, universal theories of cognition, workbench

1. Introduction

One long-standing aspiration of cognitive science is that education would benefit from the building of learning theories that are expressed, at least partially, as Artificial Intelligence (AI) programs. I have built several such programs [34,38], and others have built many more [1,19,11,5,21]. Although such work has profoundly changed our image of competence and intelligence, and that change has begun to seep into the educational system, it is fairly clear now that the resulting programs/theories have not had as much *direct* effect on education and training as could be desired. This paper examines the reasons why and suggests a new research direction based on that analysis.

The basic problem is that there seems to be an unavoidable tradeoff between the generality of learning theories and their utility to educators. Let us examine this tradeoff by starting with some recent general theories of learning and seeing what utility they have for education.

SOAR [19,16] and ACT* [1,2] aim to be universal theories of cognition. Their goal is to describe only the aspects of skill acquisition that are common to the acquisition of *all* skills. These theories are well suited for some purposes. Some examples are:

— explanations of speed and error patterns in transcription typing [12],
— explanations of the power-law increase in speed and accuracy that invariably accompany extensive practice. [24,1],
— explanations of transfer, as measured by savings in learning time caused by prior training on a similar skill [29,30,13].

However, the mechanisms of ACT* and SOAR do not in themselves tell us much about the students' *initial* acquisition of the skill. For instance, they do not tell us how students will read an instructional text, nor the effects of examples, nor the impact of specific pre-existing conceptual knowledge, nor the importance of having mental models in task domains that admit them, and so forth.

This is not an oversight on the part of the authors of ACT* and SOAR, but arises from the fact that initial acquisition of a skill seems to be a form of problem solving. Students, while engaged in various pedagogical activities such as studying a text or working some exercises, occasionally discover that their knowledge is incomplete or mistaken. This is a problem. They know many methods for solving the problem of ignorance, and different students may know different methods[1].

As always in problem solving, the behaviour of the subjects is determined mostly by the nature of the problem and the particulars of their knowledge. Neither of these is specified by ACT* or SOAR, as they aim to describe only the *universal* aspects of cognition. However, ACT* and SOAR should be consistent with the observed behaviour in that one should be able to specify (as ACT* or SOAR programs) a model of the individual subjects' knowledge and the task environment that will cause the architectures to accurately simulate his or her behaviour. Presumably, the particulars of ACT* and SOAR put some constraints on the specification of the knowledge, but the constraints imposed by the nature of the task are much stronger.

To put it differently, suppose an educator who is interested in teaching thermodynamics is not sure which of several ways of learning is typically used by thermodynamics students or could potentially be used by them. Trying these various options out on ACT* and SOAR will not reduce the educator's uncertainty one bit, because the architectures will probably be consistent with all learning methods the educator is likely to consider. In short, because these architectures aim at universality, they turn out to pretty useless as constraints on task-specific theories of initial skill acquisition.

To put the same point a third way, one view of pedagogy [4] is that a sufficient teaching method (but not, of course a necessary one) is to:

1. formalize as production rules (or some other type of rule) exactly what the students need to know in order to perform competently, and
2. design a curriculum whose lessons introduces these rules in small batches (cf. VanLehn, 1983) and
3. design lessons that explain the rules clearly and provide sufficient practice on applying them. (Immediate feedback is seen as particularly important for catching misunderstandings and rectifying them, but it is not essential to this method.)

The critical step in this teaching method is the task analysis that takes place in the first step. Task analysis is driven almost exclusively by the subject matter of the task domain. General cognitive theories, such as ACT*, provide a notation for the rules, but otherwise offer little guidance to the person conducting the analysis.

2. The essential problem, and three possible solutions

These deficiencies are not a fault of ACT* and SOAR per se. Rather, it seems that very little of our cognitive behaviour (as opposed to more peripheral behaviours) is determined by the fixed, unchangeable parts of our mind. Cognitive behaviours seem to be determined by our knowledge and the environment itself. Moreover, knowledge acquisition is a cognitive behaviour, which is itself determined mostly by knowledge and the environment. To put it in more traditional terms, because we humans are a highly adaptive species (i.e., we mould our behaviour to fit the environment), our higher level behaviour is determined mostly by our history of interaction with the environment (our knowledge) and by the environment at hand.

Unpacking the recursion here, it seems that the ultimate determinant of cognitive behaviours is the person's environment. (This is, of course, a gross simplification -- I am not proposing a tabula rasa here.) Presumably, one could explain cognitive behaviour by omitting descriptions of the various cycles of knowledge acquisition, etc. and just examine the relationship between the environment and cognitive behaviour[2]. Although this is one logically possible way to predict human behaviour, I suspect that such an explanation would be cumbersome and inaccurate, so I would not recommend pursuing it.

Logically, the only other option is to incorporate the environment into the theory. Thus, for example, a theory of physics learning would include task-specific terms like "forces" and "equations." Such theories blend psychology and the particulars of a task domain. In order to illustrate the notion of task-specific theories, let us examine some simple ones. The task of arithmetic calculation is fairly well understood. It divides cleanly into recall of arithmetic facts, such as 17-9=8, and execution of arithmetic algorithms, such as the algorithm for subtracting two multi digit numbers. We will consider a task-specific theory for recall and a task-specific theory for execution.

Siegler [26-28] has developed specific models of how students "recall" arithmetic facts. Each model has parameters that can be fit to a individual subject's behaviour, thus providing both a test of the models and a way to forecast the subject's behaviour. Each model is specific to one type of arithmetic operation, but they are all consistent with his general theory of strategy selection, which features a specific procedure for trading off retrieval and reconstruction of the item to be recalled. Reconstruction, in this context, might consist of using counting to generate

[1] Some types of problems occur so often that their solution has become routine, and subjects hardly notice that they have found and rectified a point of ignorance. For instance, students might not initially understand the referent of a mathematical symbol while reading a text or example, but after a few second's reflection, they retrieve (or construct) its meaning, and continue their reading. Presumably, they learn something from such an experience. The experience can be analyzed as a brief episode of problem solving, even though the subjects may not have thought of it as such.

[2] This proposal is similar to Anderson's Rational Analysis [3], except that the time scales and phenomena are different. Anderson seeks to explain the fixed, unchanging part of a person's mind -- the cognitive architecture -- by assuming that it is the product of genetic adaptation to the demands of the environment. The proposal here is to explain an individual's knowledge as the product of adaptation to the environment that has been experienced since birth.

an addition fact. Moreover, the general theory specifies how memory traces are strengthened by practice, thus leading to the dominance of memory retrieval over reconstruction that characterizes the competent student's performance. Siegler's theory of recall seems quite general, for it has been successfully applied to analyze acquisition of spelling rules (Siegler, personal communication) as well as the major arithmetic operations. Of course, it is not as general as ACT* or SOAR, but it serves nicely as a simple illustration of the difference between a general theory, a task-specific theory/model (e.g., the model for addition, which has explicit reconstruction strategies for arithmetic facts), and a subject-specific model (the addition model, with its parameters fit to a given subject's data). Siegler's task-specific models are specific enough that one can envision designing a curriculum around them, and Siegler has recently begun to do just that (Siegler, personal communication).

My colleagues and I have developed models of the algorithms for multi digit arithmetic, concentrating especially on subtraction [7,33,36,40]. There is a general theory, which distinguishes between normal execution of a procedure and "error handling." According to the theory, when people reach an impasse, perhaps because their knowledge of the procedure is incomplete and they can not decide what to do next, they treat the impasse itself as a problem and attempt to resolve it. One impasse-resolving strategy is to ask for help or to consult a textbook. Another is to search through one's earlier work looking for an inadvertent error. These strategies depend strongly on the particulars of situation that the students are in and on their knowledge of the task domain. Another hypothesis of the general theory is that learning occurs whenever the resolution of an impasse is summarized and stored in memory as a new rule [38]. The general theory has been tested by developing a task-specific theory/model of [33,38]. The model has been fit to individual subjects' error data. The task-specific model makes predictions about pedagogies for subtraction, some of which have been tested [35]. This work again illustrates the difference between a general theory, which offers little specific guidance to educators, and task-specific theories/models, which provide crisp suggestions.

Neither of the "general" theories just mentioned are as general as ACT* or SOAR, so a better view of the world is to see theories as arranged in some kind of generalization hierarchy. SOAR, for instance, is a straightforward generalization of both Siegler's theory and mine, because it generalizes the notion of an "impasse" to cover both failures due to memory retrieval and failures due to flawed knowledge. On the other hand, SOAR offers even less guidance to educators than either Siegler's theory or mine, just because it has more generality. So the same generality-power tradeoff is evident, even though the binary distinction between general theories and task-specific ones has dissolved into a generalization hierarchy. Although I will continue to speak of "general" versus "task-specific" theories, one should keep in mind that this is a simplification.

It seems that task-specific theories offer a viable option for guiding pedagogy. But unfortunately, task-specific theories offer little help to people who are interested in other tasks (or at least, that is how the theories are treated: theories of arithmetic are pretty much ignored by everyone except those interested in arithmetic). Thus, while task-specific theories are much more helpful to some educators than general theories, they are not helpful to very many educators.

This leads to a third option (the first two were environmental theories and task-specific theories), which is to formulate a *method* for generating task-specific theories. Traditionally, a method is a prescription of the kinds of experiments to run, the kinds of analyses to make and the kinds of conclusions to draw. The later two items are actually a weak task-general theory. It is weak because it does not foreordain the conclusions, but merely provides some ideas or even some notations for stating the task-specific theory. To put it differently, a method provides (1) a general theory and (2) a means of instantiating the theory to fit a task domain, thus formulating a task-specific theory.

There are methods in education, but I believe it is fair to say that all of them are oriented towards prescribing instruction rather than constructing learning theories. The social sciences contain many descriptive methods, such as factor analysis and its associated theory of

intelligence, or structural linguistics and its associated theory of syntax. However, as far as I know, there is no method for formulating task-specific theories of learning.

This does not bode well for a project aimed at formulating such a method. All the arguments presented above depend only on ancient concepts, such as the distinction between knowledge and its application. These arguments lead more or less inevitably to the project of formulating a method. Surely someone in the long history of education and psychology must have tried to formulate such a method. Maybe they tried and failed. Maybe such a method is just not feasible.

Some recent results in AI indicate that a method for formulating task-specific theories may indeed be feasible. Most of the work is aimed at replicating the reasoning processes behind human scientific discovery [17,25]. Although there is no denying that these programs produce the same hypotheses and experimental demonstrations that the human scientists did, there are still grave doubts about whether the simplifications assumed by these models are too strong. Pessimists would say that the machine discovery programs are not particularly intelligent, but the people who chose the simplifications for them were very intelligent. Since the pessimists could turn out to be right, it is prudent for those who wish to apply this new machine discovery technology to assume that a practical machine discovery system has a scientist/user who selects the simplifications and oversees the machine's reasoning. To put it crudely again, although the machine discovery work may or may not be able to build a mechanical scientist, it probably can build a mechanical research assistant. Such a tool could play a key role in a method for formulating task-specific theories of learning.

In short, it seems that the most promising option for finding theories of learning that are really useful to educators is to formulate a method that combines the talents of people and machine discovery programs in order generate task-specific theories of learning. This is a research option that I think should be pursued.

3. Workbenches: existing and proposed

Calling the research product a "method" makes it sound like a step-by-step prescription of how to construct a theory. I do not think *that* kind of method is feasible. What I have in mind is a set of integrated computer-based tools for analyzing data and building models. Such a "scientist's workbench" would be based on some task-general theory, such as ACT* or SOAR, or perhaps some moderately general theory, like Siegler's or mine. This section discusses some examples.

CIRRUS [39,14] is a workbench based on my theory about how people execute cognitive tasks. In addition to the hypotheses mentioned above, the theory includes the hypotheses that people are free to pick any goal that they can recall as the next goal to attend to, and their knowledge includes some policies concerning what types of goals to attend to in what situations [40][3]. CIRRUS is designed for analyzing protocol data within the framework of the theory by building a runnable simulation and comparing its behaviour to the given protocol[4]. Students' policies about goal selection are formalized as a set of goal selection preferences of the form "If condition C holds, then prefer goals of type A over goals of type B." The simulator uses such preferences to sort a list of pending goals and choose the goal that is preferred above all others. To use CIRRUS, the theorist must input a procedure, written in the knowledge representation language of the theory, that lacks goal selection preferences. CIRRUS must also be given primitives from which goal selection preferences can be built. Given a protocol, CIRRUS builds goal selection preferences that allow a maximally accurate simulation of the data. To put it in more traditional terms, CIRRUS takes a model with one parameter, and fits it to the given data. However, both the model and the parameter are non-numeric.

[3]This theory is slightly more general than ACT* and SOAR. Those theories claim that people invariably select one of the unsatisfied goals that was created most recently (i.e., both ACT* and SOAR have a last-in-first-out goal stacks)

[4]CIRRUS does not understand natural language; the protocol must be encoded by humans before giving it to CIRRUS.

When my collaborators and I use CIRRUS, we find it necessary to refine the model given to it many times before we are finally happy with the analysis it yields. Typically, we analyze one subject's data in some detail, then start our analysis of the next subject using the model developed for the first subject. After several subjects have been analyzed, commonalties in the subject-specific models emerge. At that point, we build a subject-general model and install parameters (typically, a system of switches that turn rules off and on) in order to capture the between-subjects variation. We stop the analysis when all the subjects have been analyzed and one subject-general model has been found. One of the model's parameters, the set of goal-selection procedures, is fit automatically by CIRRUS; the other parameters, which were created during the model refinement process, are fit by hand. This refinement process can be viewed as finding a theory that is specific to the task under analysis but general across subjects. In this fashion, CIRRUS helps the scientist/user discover a task-specific theory/model.

ACM [22,23] is similar to CIRRUS. It is based on the theory that problem solving is search through a problem space. It takes as its model a specific problem space, and builds a set of operator selection heuristics that will cause search through this problem space to simulate answer data given to the program.

SAPA [6] is somewhat like ACM, in that it is based on the theory of problem solving as search through a problem space. However, it does not to actually build a set of search heuristics that fit some data given to it. It already has some search heuristics in it, along with a particular problem space[5]. These search heuristics are intended, I suppose, to represent those of a prototypical subject's. At each cycle of the search, SAPA asks the user if the inference it has just made corresponds to the protocol. If it does, then the built-in, fully parameterized model is upheld. If not, then SAPA checks to see if the parameterization is wrong -- i.e., it has the right problem space but the wrong heuristics for that subject. It performs this check by suggesting alternatives until the user indicates that it has found one that corresponds to the protocol. If none of SAPA's suggestions work, then the problem space is deemed faulty, because no parameterization of the model will fit the data. Bhaskar and Simon used SAPA to test their task-specific theory of thermodynamics problem solving, and to test their model of a prototypical student's search heuristics.

All these workbenches, as well as several others (e.g., Debuggy [8], TETRAD [10] and METADENDRAL [18] have three components: (1) a general theory that is so deeply embedded in the workbench that it can not be changed, (2) a underdetermined model given to the workbench by the user, such as a problem space for thermodynamics problem solving, and (3) a process that fits the model to the data, making it more deterministic. The theorist tinkers with the underdetermined model in order to get a fitted model that analyzes the data satisfactorily. The result is a model that is both a generalization over several (hopefully, many) subjects' data and a specialization of the general theory. The model can be considered a task-specific theory.

Of course, such a model is interesting only to the extent that that task is interesting. Educators are interested in learning, but CIRRUS, ACM and SAPA all assume that learning does not occur during the protocols they are analyzing. Thus, they could be used in a longitudinal study to model snapshots of the learner's development, but they can not model the learning process itself. This leads to a proposal to build a workbench that can model the learning process.

I am currently involved in building a scaled-up version of CIRRUS, called CASCADE. CASCADE is being built in order to analyze a very large data set, donated by Micki Chi[9]. The data consist of 8 protocols, each about 200 pages long. They were collected from students studying the first four chapters of a college physics textbook. The protocols record the learning that a typical college student would undergo in the first few weeks of a college physics course.

[5] Although SAPA was build to handle only thermodynamics, it could be redesigned to have more task generality by allowing the user to input a problem space.

4. Expected benefits of the proposed research

The most important application of the proposed technology is providing a "front end" to projects that create training systems. According to Anderson, the first step in developing a training system is to analyze the task domain to see what good students should know when they have completed their training [4]. Workbenches such as CASCADE are intended to help a designer perform such a task analysis. Although this section suggests a few other benefits that might accrue, one should keep in mind that the main benefit is technological assistance in task analysis.

The task-specific, subject-general model that is created on the workbench could be the starting point of the development of a student modeller for an intelligent tutoring system. Also, the data analysis tools developed as parts of the workbench could be used as parts of the diagnostic module of an intelligent tutoring system.

The mere process of analyzing students' learning in the face of the given instructional material will usually reveal defects in the material that can be easily remedied. Anderson, for instance, has a written a textbook on LISP based on his task analysis. Since the analysis had only got as far as recursion when the book was written, the last five chapters in the text were not based on a task analysis. Anderson comments: "Since the writing of the book we have slowly began to create tutor material corresponding to those chapters. As we have done so we have started to realize the inadequacy of the information in the last five chapters." [3 ch. 4]. It is significant that task analysis of the initial segment of the curriculum, even by someone like Anderson, was not sufficient preparation for writing an adequate material for the second segment. It seems that there is no substitute for formal task analysis, even if the intended training vehicle is "just" a textbook.

Once a task-specific model of the student has been constructed, it often suggests new pedagogical strategies. Given the model, some will seem clearly beneficial. However, pedagogies whose benefits are less certain can be simulated; if the model is psychologically accurate, and the proposed benefit helps the model learn, then human students should learn better as well. For instance, on the basis of Siegler's model of addition, it seems that under certain circumstances, supervised drill can take advantage of the commutativity of addition and only teach half the addition facts. Unsupervised drill on the other half should suffice for learning them. This pedagogical regime should be tested on his model before being tried in the classroom.

So far, the importance of this work to education has been stressed. But there are other potential beneficiaries as well. Machine learning has recently turned towards scientific discovery as a source of new problems. Because a workbench is a program that participates in scientific discovery, it should be of some interest to research on discovery. One can even imagine taking protocols of scientists while they use it in order to understand the discovery-making process better.

In protocols of students involved in learning new material, such as the ones being analyzed by the CASCADE project, there are many instances of students making discoveries. These discoveries might suggest discovery methods that could be developed into full-fledged machine learning techniques.

Looking further ahead, machine learning has not yet produced interactive learners that can hold up their end of a training dialogue with their trainer. Formal work in the Valiant framework ("PAC learning") indicates that such interactivity is necessary for tractable learning [31], so eventually machine learning will have to build such interactive learners if it is to live up to its promises of delivering systems that acquire knowledge for expert systems. The current protocol studies show how interaction proceeds with human students. That should suggest styles of interaction to machine learning researchers.

Turning now to the benefits for psychology, we start with the traditional observation that applications usually push theories towards completion because application efforts do not have the luxury of ignoring parts of human behaviour that are difficult to explain. This application of cognitive theory will certainly push it towards completion. For instance, the physics task domain is richer in conceptual material than other task domains, such as LISP and geometry, that have been studied. Thus, the development of a task-specific theory in physics should illuminate the interaction between conceptual and procedural learning.

I have concentrated on workbenches for analyzing protocol data because such data will push cognitive theory along by explicating the mapping between theoretical events, such as impasses, and visible types of human behaviour. There are few published comparisons of protocols and models as detailed as the analyses in Human Problem Solving [20], and none that compare models and students who are learning. The CASCADE project, and others like it, should yield the first fine-grained analysis of human learning. From such analyses, we ought to uncover some unexpected theoretical problems, as well as strengthen known weak spots in the theory.

References

1. Anderson, J. R. *The Architecture of Cognition*. Cambridge, MA: Harvard. 1983.
2. Anderson, J. R. Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review*, 94(2), 192-210. 1987.
3. Anderson, J. R. *The Adaptive Character of Thought*. Cambridge, MA: MIT Press. (in press).
4. Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. J. Cognitive principles in the design of computer tutors. *In Proceedings of the Cognitive Science Society Conference*. Hillsdale, NJ: Erlbaum. 1984.
5. Anzai, T. Doing, understanding and learning in problem solving. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production System Models of Learning and Development*. Cambridge, MA: MIT Press. 1987.
6. Bhaskar, R. & Simon, H. A. Problem solving in a semantically rich domains: An example from engineering thermodynamics. *Cognitive Science*, 1, 193-215. 1977.
7. Brown, J. S. & VanLehn, K. (1980). Repair Theory: A generative theory of bugs in procedural skills. *Cognitive Science*, 4, 379-426.
8. Burton, R. B. Diagnosing bugs in a simple procedural skill. In D. H. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*. New York: Academic. 157-183. 1982.
9. Chi, M. T. H., Bassok, M., Lewis, M., Reimann, P., & Glaser, R. Self explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13, 145-182. 1989.
10. Glymour, C., Scheines, R., Spirtes, P., & Kelly, K. *Discovering Causal Structure*. Orlando, FL: Academic Press. 1987.
11. Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. *Induction: Process of Inference, Learning and Discovery*. Cambridge, MA: MIT Press. 1986.
12. John, B. *Contributions to engineering models of human-computer interaction*. Doctoral dissertation, Department of Psychology, Carnegie Mellon University. 1988.
13. Kessler, C. M. *Transfer of Programming Skills in Novice LISP Learners*. Doctoral dissertation, Department of Psychology, Carnegie Mellon University. 1988.
14. Kowalski, B. & VanLehn, K. Inducing subject models from protocol data. In V. Patel (Ed.), *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum. 1988.
15. Kulkarni, D. & Simon, H. A. The process of scientific discovery: the strategy of experimentation. *Cognitive Science*, 12, 139-175. 1988.
16. Laird, J. E., Newell, A., & Rosenbloom, P. S. Soar: an architecture for general intelligence. *Artificial Intelligence*, 33, 1-64. 1987.
17. Langley, P., Simon, H. A., Bradshaw, G. L, & Zytkow, J. M. *Scientific Discovery: Computational Explorations of the Creative Process*. Cambridge, MA: MIT Press. 1987.
18. Lindsay, R., Buchanan, B. H., Feigenbaum, E. A., & Lederberg, J. *DENDRAL*. New York: McGraw-Hill. 1980.
19. Newell, A. *Unified Theories of Cognition*. Cambridge, MA: Harvard. (in press).
20. Newell, A. & Simon, H. A. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall. 1972.
21. Ohlsson, S. Truth versus appropriateness: relating declarative to procedural knowledge. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production System Models of Learning and Development*. Cambridge, MA: MIT Press. 1987.
22. Ohlsson, S. & Langley, P. *Identifying solution paths in cognitive diagnosis*. (Technical Report CMU-RI-TR-84-7). Pittsburgh, PA: Robotic Institute, CMU. 1985.
23. Ohlsson, S. & Langley, P. Psychological evaluation of path hypotheses in cognitive diagnosis. In H. Mandl & A. Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems*. New York: Springer Verlag. 1988.
24. Rosenbloom, P. & Newell, A. Learning by chunking: A production system model of practice. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production System Models of Learning and Development*. Cambridge, MA: MIT Press. 1987.
25. Shavlik, J. W. Learning classical physics. In T. M. Mitchell, J. G. Carbonell, & R. S. Michalski (Eds.), *Machine Learning: A Guide to Current Research*. Higham, MA: Kluwer. 1986.
26. Siegler, R. S. Strategy choices in subtraction. In J. Sloboda & D. Rogers (Eds.), *Cognitive Processes in Mathematics*. Oxford, UK: Oxford University Press. 1987.
27. Siegler, R. S. Strategy choice procedures and the development of multiplication skill. *Journal of Experimental Psychology: General*, 117, 258-275. 1988.
28. Siegler, R. S. & Shrager, J. Strategy choices in addition: How do children know what to do? In C. Sophian (Ed.), *Origins of Cognitive Skill*. Hillsdale, NJ: Erlbaum. 1984.
29. Singley, M. K. & Anderson, J. R. The transfer of text-editing skill. *International Journal of Man-Machine Studies*, 22, 403-423. 1985.
30. Singley, M. K. & Anderson, J. R. A keystroke analysis of learning and transfer in text editing. *Human-Computer Interaction*. 1988.
31. Valiant, L. G. A theory of the learnable. *Communications of the ACM*, 27(11), 1134-1142. 1984.
32. VanLehn, K. *Felicity conditions for human skill acquisition: Validating an AI-based theory* (Technical Report CIS-21). Xerox Palo Alto Research Center. Out of print, but available from University Microfilms, 300 North Zeeb Road, Ann Arbor, MI 49106. 1983a.
33. VanLehn, K. Human skill acquisition: Theory, model and psychological validation. In M. R. Genesereth (Ed.), *Proceedings of AAAI-83*. Los Altos, CA: Morgan Kaufmann. 1983b.
34. VanLehn, K. Learning one subprocedure per lesson. *Artificial Intelligence*, 31(1), 1-40. Reprinted in Shavlik, J. W. & Dietterich, T. G. (Eds.). *Readings in Machine Learning*. Palo Alto, CA: Morgan Kaufmann. 1987.
35. VanLehn, K. Toward a theory of impasse-driven learning. In H. Mandl & A. Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems*. New York: Springer Verlag. 1988.
36. VanLehn, K. *Mind Bugs: The origins of procedural misconceptions*. Cambridge, MA: MIT Press. 1990.
37. VanLehn, K. Felicity conditions for cognitive skill acquisition: Tutorial instruction does not need them. *Cognition and Instruction*. Currently available as technical report PCG-17, Department of Psychology, Carnegie Mellon University.(submitted).
38. VanLehn, K. & Ball, W. Teton: An architecture for analysis of protocol data. In K. VanLehn (Ed.), *Architectures for Intelligence*. Hillsdale, NJ: Erlbaum. (in press).
39. VanLehn, K. & Garlick, S. Cirrus: An automated protocol analysis tool. In P. Langley (Ed.), *Proceedings of the Fourth Machine Leaning Workshop*. Los Altos, CA: Morgan Kaufmann. 1987.
40. VanLehn, K., Ball, W., & Kowalski, B. Non-LIFO execution of cognitive procedures. *Cognitive Science*, 13, 415-465. 1989.