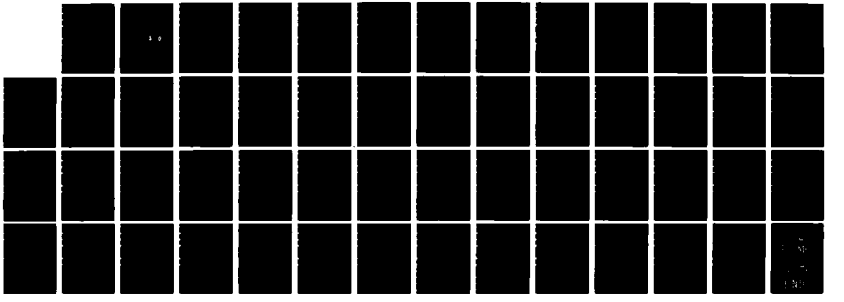
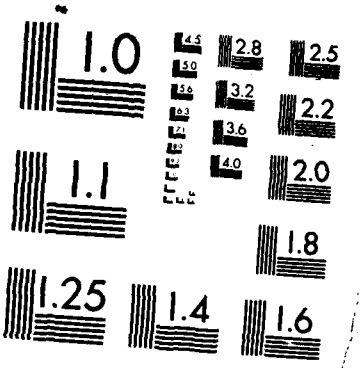
AD-A164 588 ACQUIRING PROCEDURAL SKILLS FROM LESSON SEQUENCES(U) 1/1
XEROX PALO ALTO RESEARCH CENTER CA K A VANLEHN
13 AUG 85 ISL-9 N00014-82-C-0067

UNCLASSIFIED F/G 5/10 NL





Acquiring Procedural Skills From Lesson Sequences

Kurt VanLehn

July 1985

ISL-9

[P85-00100]

XEROX

PALO ALTO RESEARCH CENTERS
3333 Coyote Hill Road / Palo Alto / California 94304

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

AD-A164 580

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ISL-9	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Xerox Palo Alto Research Center 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Personnel and Training Research Program Office of Naval Researh (Code 442 PT)
6c. ADDRESS (City, State, and ZIP Code) 3333 Coyote Hill Road Palo Alto, CA 94304	7b. ADDRESS (City, State, and ZIP Code) Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-82C-0067
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 61153N; PROJECT NO: RR042-06; TASK NO: RR042-06-0A; WORK UNIT ACCESSION NO: NR667-477

11. TITLE (Include Security Classification)
Acquiring Procedural Skills From Lesson Sequences

12. PERSONAL AUTHOR(S)
Vanlehn, Kurt Alan

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 1/1/82 TO 6/15/85	August 13, 1985	52

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Learning, skill acquisition, cognitive science, procedural skills, bugs, felicity conditions, repairs.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report provides an informal overview of a theory that describes how people learn certain procedural skills, such as arithmetic and algebra, from multi-lesson curricula. The central hypothesis is that students and teachers obey conventions that cause the goal hierarchy of the acquired procedure to be a particular structural function of the sequential ordering of lessons. This learning theory is an extension of Repair Theory, which describes how people mix interpretation and a certain type of meta-level problem solving as they try to solve practice problems. The learning theory has been embedded in a program that generates detailed predictions about the products of published curricula. The predictions have been tested against data from several thousand mathematics students.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Mike Shafto	22b. TELEPHONE (Include Area Code) (202) 696-4322
	22c. OFFICE SYMBOL 4 42 PT

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

Acquiring Procedural Skills From Lesson Sequences

Kurt VanLehn

Abstract

This document is the final report for ONR contract N00014-82C-0067. It provides an informal overview of a theory that describes how people learn certain procedural skills, such as arithmetic and algebra, from multi-lesson curricula. The central hypothesis is that students and teachers obey conventions that cause the goal hierarchy of the acquired procedure to be a particular structural function of the sequential ordering of lessons. This learning theory is an extension of Repair Theory, which describes how people mix interpretation and a certain type of meta-level problem solving as they try to solve practice problems. The learning theory has been embedded in a program that generates detailed predictions about the products of published curricula. The predictions have been tested against data from several thousand mathematics students.

Acknowledgments

I would like to thank the readers whose thoughtful comments have helped this document along: John Seely Brown, Tom Moran, Lissa Monty and Jeff Shrager. This research was supported by the Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research, under Contract Number N00014-82C-0067. Contract Authority Identification Number NR667-477. Reproduction in whole or in part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

Acquiring Procedural Skills From Lesson Sequences

Kurt VanLehn

The research presented here began with the "buggy" studies of Brown and Burton (1978). Those studies found that students of certain procedural skills, such as ordinary multicolumn subtraction, had a surprisingly large variety of *bugs* (small, local misconceptions that cause systematic errors). Early investigations into the origins of bugs yielded a theory of procedural problem solving, Repair Theory (Brown & VanLehn, 1980). A subsequent empirical study (VanLehn, 1982) confirmed many of Repair Theory's predictions, including the surprising prediction that certain bugs would be replaced by others during a short periods of time, a phenomenon called *bug migration*. Recent research has investigated the relationship between the curriculum, the students' learning processes and the acquisition of bugs. A learning theory has been added to Repair Theory, yielding an integrated explanation for the acquisition of correct and buggy procedures (VanLehn, 1983).

This article provides an introduction to the learning theory. It omits as much detail as possible in order to concentrate on the theoretical and methodological intuitions that underlie the theory. In particular, it omits the empirical arguments that support the theory's hypotheses. Facts about student behavior are sprinkled throughout, but are used merely to illustrate the theory's claims. Proper arguments for a theory of this complexity require a book (e.g., VanLehn, forthcoming-a) to present them.

The article begins with a discussion of methodological goals of the research. The middle sections introduces the main hypotheses of the theory. The final section outlines the validation methods.

1. Eliminating of the program parameter

Artificial Intelligence has always had difficulty validating the models of cognition that it proposes (VanLehn, Brown & Greeno, 1984). This is due, in part, to the complexity of those models. Recently, increasing computer power has made it feasible for programs to *learn* how to do complex tasks, and it is much easier to validate a learning program than a program that does not learn. This may seem counterintuitive, since learning programs are generally more complicated than

non-learning programs. Yet validating a learning model is easier because it avoids an important methodological problem, which I call *the program parameter problem.* This problem is complex and subtle. The following treatment of it is at best a mere gloss of the issues involved. A thorough discussion can be found in Pylyshyn's excellent book (Pylyshyn, 1984).

The program parameter problem occurs when a model must be given a complicated expression, written in a formal representation language, in order to simulate a given task. It is appropriate to call the expression a program because the actions of the model are determined by interpreting the expression. This is true regardless of whether the expression is a procedural encoding of knowledge or a declarative encoding. From a methodological point of view, the program is a parameter of the model, although a powerful and multi-faceted one. So the defining characteristic of the models under discussion is that they take a program parameter. The following examples illustrate this concept. Newell (1978) proposes a certain production system architecture as a model of the mind. To parameterize it for a given task, the theorist provides a list of productions. The production system's speed is intended to correlate with the subject's speed when they are given the same problems to solve. For a second example, Collins and Loftus (1975) propose a spreading activation architecture for semantic memory. It is parameterized by writing a semantic net in a representation language. The time required for activation to spread through the given net is intended to be proportional to the speed with which subjects answer questions. In both these examples, the model of cognition has a program parameter: productions in the first example; a semantic net in the second example.

When a model has a program parameter, it almost always has two undesirable characteristics. First, small changes in the value of the program parameter (i.e., a slightly different program) can cause significant changes in the predictions made by the model. That is, the model is extremely sensitive to the value of its parameter. If one assumes that all possible programs are, a priori, equally probable, then the theory must explain why one particular program is the only one that works. Thus, the model has converted a hard problem, such as explaining why people solve problems or answer questions at certain speeds, into an even harder problem: explaining why they have a certain program for that task.

The second undesirable characteristic of program parameter models is that it is almost always the case that one can devise a new model, convert the old program into the appropriate format for the new model, and get equally accurate preditions. For instance, Newell (1973) and Newell (1978) proposed different production system

architectures, but get roughly the same accuracy for a certain task (the Sternberg task). To take a new example, Newell and Simon (1972) propose a certain cognitive architecture and demonstrate that it can be programmed to accurately simulate long, complicated protocols of subjects solving difficult puzzles. However, it is clear that one could re-write their programs to run on an implausible cognitive architecture, e.g., Pascal, and still produce an accurate simulation of the subjects' performance. This indicates that the predictive accuracy of the model as a whole depends entirely on the program parameter's value (i.e., a certain program). One gets equivalent predictions by substituting various architectures while keeping the same program (i.e., the same value for the parameter).

A cognitive model that *learns* how to solve a task does not need a program parameter. It constructs (learns) the program that a theorist would otherwise have to provide. Although such a model has no program parameter, it does have one input, a formal expression that stands for the training and/or instructions that the subjects' received. However, this input is not a parameter, because its value can be *observed.* It is an independent variable, not a parameter.

Because models of learning lack program parameters, they are much easier to validate. If the model is making successful predictions, then one must credit the model because there are no program parameters to steal the credit from the model. The problem of explaining why program parameters have certain values and not others does not exist for independent variables. They have the values they do because those values are proper encodings of certain observable facts.

On the other hand, computer models of learning are much harder to build. Worse, they have a tendency to run quite slowly and use large amounts of memory. Only in recent years has it become feasible to construct and debug models of non-trivial learning. Even so, such models are difficult to work with. For instance, the learning model described herein is implemented as a Lisp program named Sierra. Sierra takes a week of computer time (i.e., 150 cpu-hours on a Dorado, which is one of the fastest personal Lisp machine currently available) to do one run, where a run consist of learning a skill from a lesson sequence while generating predictions about the subjects' problem-solving behavior at various points along the way. Many runs have been made, both to debug Sierra and to try out various versions of the model in order to see which ones produced the most accurate predictions. The amount of computer time required for such testing was simply unavailable a decade ago.

Trying out various versions of the model contributes significantly to the cpu usage, but it is essential for moving beyond a mere demonstration that Sierra is *sufficient* to predict the data. In order to converge on a demonstration that Sierra (or rather, a class of Sierra-equivalents) is *necessary* for accurate prediction, many versions of the program must be tried with alternative, competing hypotheses substituted for the hypotheses that the model/theory subscribes to. The importance of moving from sufficiency to necessity is discussed in section 5, and more fully in VanLehn, Brown & Greeno (1984) and Pylyshyn (1984)

Increasing computer power sets the stage for a new era in cognitive science where complex cognition, the kind that AI has speculated about, can be studied empirically and rigorously. The key is to eliminate program parameters from cognitive models by studying not only how a complex skill is performed, but how the skill is acquired as well.

2. Learning elementary mathematical skills

The goal of this research is to develop an rigorously supported theory of learning by taking advantage of AI's new modelling power. The long term research strategy is to begin by studying a particular kind of cognition, then if all goes well, to test the theory's generality on other kinds of cognition. The initial studies focused on how elementary school students learn ordinary, written mathematical calculations.

The main advantage of mathematical procedures, from a methodological point of view, is that they are virtually meaningless to most students. They seem as isolated from common sense intuitions as the nonsense syllables of early learning research. In the case of the subtraction procedure, for example, most elementary school students have only a dim conception of its underlying semantics, which is rooted in the base-ten representation of numbers (VanLehn & Brown, 1980; VanLehn, 1983; VanLehn, 1985b). When compared to the procedures students use to operate vending machines or play games, arithmetic procedures are as dry, formal and isolated from everyday interests as nonsense syllables are different from real words. This isolation is the bane of teachers, but a boon to psychologists. It allows psychologists to study a skill that is much more complex than recalling nonsense syllables, and yet it avoids bringing in a whole world's worth of associations. Given the methodological goal of a zero-parameter model, this is essential. If a skill were chosen that did require significant prior knowledge, then that knowledge might have to be represented as a program parameter.

The remainder of this section introduces the domain. First it describes the instruction that students receive, and then it describes the behavior they produce. The theory's main job is to explain what kinds of mental structures are engendered by that instruction and how those structures guide the production of the observed behavior.

Learning from lesson sequences of examples and exercises

In a typical American school, mathematical procedures are taught incrementally via a lesson sequence that extends over several years. In the case of subtraction, there are about ten lessons in the sequence that introduce new material. The lesson sequence introduces the procedure incrementally, one step per lesson, so to speak. For instance, the first lesson might show how to do subtraction of two-column problems. The second lesson demonstrates three-column problem solving. The third introduces borrowing, and so on. The ten lessons are spread over about three years, starting in the late second grade (i.e., at about age seven). These lessons are interleaved with lessons on other topics, as well as many lessons for reviewing and practicing the material introduced by the ten lessons. In the classroom, a typical lesson lasts an hour. The teacher solves some problems on the board with the class, then the students solve problems on their own. If they need help, they ask the teacher, or they refer to worked examples in the textbook. A textbook example consists of a sequence of captioned "shapshots" of a problem being solved (see figure 1). Textbooks have very little text explaining the procedure (young children do not read well). Textbooks contain mostly examples and exercises.

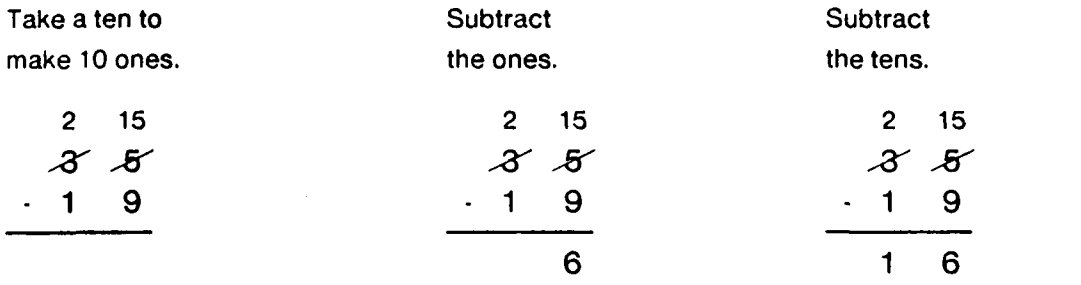


Figure 1
A typical textbook example

This brief overview of subtraction instruction illustrates (but does not validate) two important hypotheses that seem to hold for all the skills in this domain. First, skill acquisition in this domain is some kind of induction (i.e., the discovery of a general idea from examples of it). That is, procedures are learned from examples of their application. Second, inductive learning occurs in the context of an extended lesson sequence that introduces the skill incrementally. Students in the middle of the lesson sequence can be expected to have incomplete procedures that can successfully solve only some of the class of possible problems in the domain.

Describing systematic errors with "bugs"

The observable output of the students' learning process is their performance while solving exercise problems. A traditional measure of such performance is a protocol that records the student's actions in detail, including the time between actions. In this domain, the timing data is rather uninteresting. Often, students cannot remember an arithmetic fact. (In this paper, "arithmetic facts" will refer to propositions like $5+7=12$ or $7<11$.) When students forget an arithmetic fact, they count, which shows up as long pauses in the protocols. The timing data reveals more about their knowledge of arithmetic facts than their knowledge of the procedure. Since it is their procedural knowledge that is the target of this theory's explanations, error data have been used in preference to timing data.

There have been many empirical studies of the errors that students make in arithmetic (Buswell, 1926; Brueckner, 1930; Brownell, 1941; Roberts, 1968; Lankford, 1972; Cox, 1975; Ashlock, 1976). A common analytic notion is to separate systematic errors from slips (Norman, 1981). Systematic errors appear to stem from consistent application of a faulty method, algorithm or rule. Slips are unsystematic "careless" errors (e.g., facts errors, such as $7-3=5$). Since slips occur in expert performance as well as student behavior, the common opinion is that they are due to inherent "noise" in the human information processor. Systematic errors on the other hand are taken as stemming from mistaken or missing knowledge, the product of incomplete or misguided learning. Only systematic errors are used in testing the present theory. See Siegler & Shrager (in press) for a theory of addition slips.

Brown and Burton (1978) used the metaphor of bugs in computer programs in developing a precise, detailed formalism for describing systematic errors. The basic idea is that a student's errors can be accurately reproduced by taking some formal representation of a correct procedure and making one or more small perturbations to it, e.g., deleting a rule. The perturbations are called bugs. A systematic error is

represented as a correct algorithm for the skill plus a list of one or more bugs. Bugs describe systematic errors with unprecedented precision. If a student makes no slips, then his or her answers on a test exactly match the buggy algorithm's answers, digit for digit. Bug data are the main data for testing this theory.

Burton (1981) developed an automated data analysis program, called Debuggy. Using it, data from thousands of students learning subtraction were analyzed, and 76 different kinds of bugs were observed (VanLehn, 1982). Similar studies discovered 68 bugs in addition of fractions (Shaw et. al., 1982), several dozen bugs in simple linear equation solving (Sleeman, 1984), and 57 bugs in addition and subtraction of signed numbers (Tatsuoka & Baillie, 1982).

It is important to stress that bugs are only a notation for systematic errors and not an explanation. The connotations of "bugs" in the computer programming sense do not necessarily apply. In particular, bugs in human procedures are not always stable. They may appear and disappear over short periods of time, often with no intervening instruction, and sometimes even in the middle of a testing session (VanLehn, 1982). Often, one bug is replaced by another, a phenomenon called bug migration.

Mysteries abound in the bug data. Why are there so many different bugs? What causes them? What causes them to migrate or disappear? Why do certain bugs migrate only into certain other bugs? Often a student has more than one bug at a time – why do certain bugs almost always occur together? Do co-occurring bugs have the same cause? Most importantly, how is the educational process involved in the development of bugs? One objective of the theory is to explain some of these bug mysteries.

Another objective is to explain how procedural skills are acquired from multi-year curricula. This objective seems to require longitudinal data, where each student in the study is tested several times during the multi-year period. Such data is notoriously difficult to acquire. Bug data are readily available and nearly as good. Our bug data are obtained by testing students at all stages in the curriculum. Thus, the bug data are like between-subjects longitudinal data. Instead of testing the same student at several times at different stages of his or her learning, different students at different stages are tested just once. As will be seen in the next section, such data can perform nearly as well as longitudinal data in testing a learning theory, and yet they are much easier to collect.

3. An introduction to the model: Explaining Always-Borrow-Left

Most of the mental structures and processes proposed by the theory can be introduced and illustrated by going through an explanation for a certain subtraction bug, called Always-Borrow-Left. Students with this bug always borrow from the leftmost column in the problem no matter which column originates the borrowing. Problem A below shows the correct placement of borrow's decrement. Problem B shows the bug's placement.

		5			2			5
A.		3 $6^{1}5$	B.		~~3~~ $6^{1}5$	C.		$6^{1}5$
	-	1 0 9		-	1 0 9		-	1 9
		2 5 6			1 6 6			4 6

(The small numbers represent the student's scratch marks.) Always-Borrow-Left is moderately common. In a sample of 375 students with bugs, six students had this bug (VanLehn, 1982). It has been observed for years (c.f. Buswell, 1926, pg. 173, bad habit number s27). However, this theory is the first to offer an explanation for it.

The explanation begins with the hypothesis that students use induction (generalization of examples) in learning where to place the borrow's decrement. All the textbooks used by students in our sample introduce borrowing using only two-column problems, such as problem C above. Multi-column problems, such as A, are not used. Consequently, the student has insufficient information to induce an unambiguous description of where to place the borrow's decrement. The correct placement is in the left-adjacent column, as in A. However, two-column examples are also consistent with decrementing the leftmost column, as in B.

The next hypothesis of the theory is that when a student is faced with such an ambiguity in how to describe a place, the student takes a conservative strategy and saves all the relevant descriptions. When inducing from two-column problems (e.g., C), the student describes the borrow-from column as "a column that is both left-adjacent to the current column and the leftmost column in the problem."

Suppose that our student is given a diagnostic test at this point in the lesson sequence and that the test contains borrowing problems of all kinds. The student is faced with solving problem D, below.

D.		3 6 5	E.		3 $6^{1}5$
	-	1 0 9		-	1 0 9

The student starts to borrow, gets as far as E, and is suddenly stuck. The student's description of where to borrow is ambiguous because there is no column that is *both* left-adjacent and the leftmost column. In the terminology of the theory, getting stuck while problem solving is called reaching an *impasse.*

It is hypothesized that whenever students reach an impasse on a test, they engage in *local problem solving.* Local problem solving is just like classical puzzle solving (e.g., Newell & Simon, 1972), in that there is an initial state, a desired final state, and state-change operators. Here, the initial state is being stuck, and the desired final state is being unstuck. Unlike traditional problem solving, however, the state-change operators of local problem solving don't change the state of the exercise problem. Instead, they change the *state of the interpreter* that is executing the procedure. The operators do things like pop the stack of goals or relax the criterion for matching a description to the exercise problem. They do not do things like writing digits on the test paper. Because the local problem solver modifies the state of the procedure's interpretation, it is a kind of *meta-level* problem solving. The sequences of meta-level operators that succeed in getting students unstuck are called *repairs.* Note that what is being repaired is, roughly speaking, the impasse. Repairs do not change the procedure. To put it in terms of Newell's problem space hypothesis (Newell, 1980), the procedure works in one problem space, and local problem solving works in a second problem space that is "meta" to the base problem space. Returning to our stuck student, three common repairs to the impasse are illustrated below.

$$\text{F.}\quad \begin{array}{r} 2\;\;\;\;\;\;\; \\ \not{3}\;6\;{}^{1}5 \\ -\;1\;0\;9 \\ \hline \end{array} \qquad \text{G.}\quad \begin{array}{r} 5\;\;\;\; \\ 3\;\not{6}\;{}^{1}5 \\ -\;1\;0\;9 \\ \hline \end{array} \qquad \text{H.}\quad \begin{array}{r} 3\;6\;{}^{1}5 \\ -\;1\;0\;9 \\ \hline 6 \end{array}$$

In F, the student has relaxed the description of which column to borrow from by ignoring the restriction that the column be left-adjacent to the current column. The remaining restriction, that the column be the left-most column in the problem, has the student decrement the hundreds column, as shown in F. This is one repair. It generates the bug Always-Borrow-Left. Another repair is shown in G. Here, the student has relaxed the borrow-from description by ignoring the left-most requirement. The decrement is placed in the left-adjacent column, yielding G. This repair generates a correct solution to the problem. In H, the student has chosen to skip the borrow-from entirely, and go on to the next step in the procedure. This repair generates a bug that is named Borrow-No-Decrement-Except-Last, because it only executes a borrow-from when it is unambiguous where to place the decrement, and

that occurs only when the borrow originates in the last possible column for borrow. To sum up, three different repairs to the same impasse generate two different bugs and a correct version of subtraction.

It was mentioned earlier that students' bugs are not like bugs in computer programs because students' bugs are unstable. Students shift back and forth among bugs, a phenomenon called bug migration. The theory's explanation for bug migration is that the student has a stable underlying procedure, but that the procedure is incomplete in such a way that the student reaches impasses on some problems. The student can apply any repair she can think of. Sometimes she chooses one repair, and sometimes she chooses others. The different repairs manifest themselves as different bugs. So bug migration comes from varying the choice of repairs to a stable, underlying impasse. In particular, the theory predicts that the three repairs just discussed ought to show up as a bug migration. In fact, they do. Figure 2 is a verbatim presentation of a diagnostic test showing the predicted bug migration.

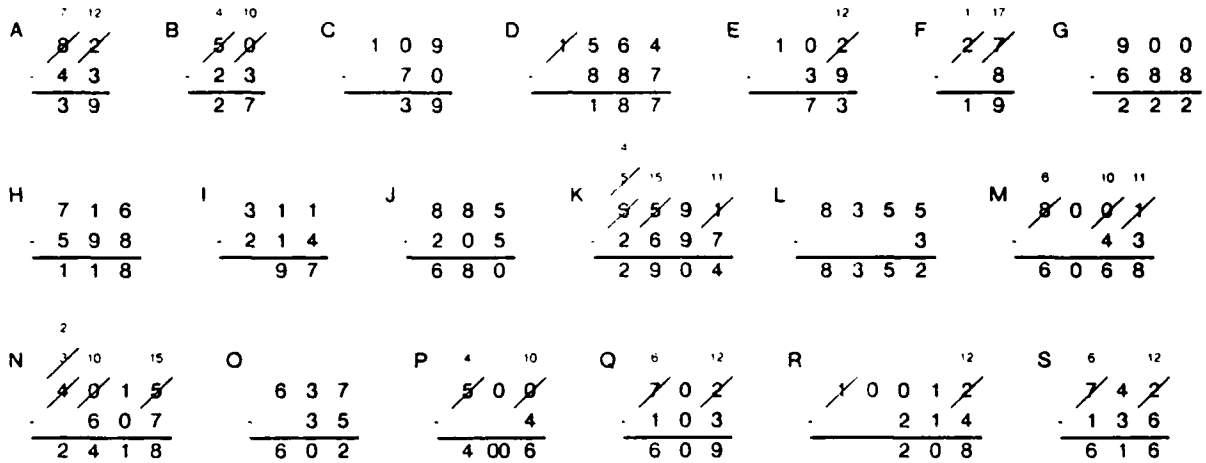


Figure 2

Verbatim presentation of a test by subject 8 of class 17 showing three repairs to the same impasse. On problems D, E and G, one repair generates the bug Borrow-No-Decrement- Except-Last. On problems H and I, another repair generates the correct borrow-from placement. On problems K, M, N, P, Q, R and S, a third repair generates the bug Always-Borrow- Left. There are slips on problems D, P, Q and S.

This discussion of the bug Always-Borrow-Left has illustrated many of the important claims of the theory. First, procedures are the result of generalization of examples, rather than, say memorization of verbal or written recipes. There are accidental, visual characteristics of the examples, viz. the placement of the decrement, that a non-example source of instruction, such as a verbal recipe, would not mention. The appearance of these visual characteristics in the acquired procedure is evidence that they were learned principally by induction (see VanLehn, 1985b, for a full defense of this idealization).

A second claim is that learning occurs in the context of a lesson sequence, and that many bugs are caused by testing students who are in the middle of the lesson sequence on exercise types that they have not yet been taught how to solve. Perhaps such bugs should be welcomed as signs of a healthy learning process that may eventuate in a correct understanding of the procedure. Such a view of bugs is radically different from the traditional view, which considers bugs to be "bad habits" that need to be remediated. On the other hand, the bad-habit view may be appropriate for older students, some of whom have bugs long after the lesson sequence has been completed (VanLehn, 1982).

Another set of claims involves the notions of interpretation, impasses and repairs. A particularly important hypothesis is that repairs occur at the meta-level and change only the state of the interpretation. This hypothesis predicts the existence of bug migration. In fact, this prediction was made before any evidence of bug migration had been found (Brown & VanLehn, 1980). The surprising success of this forecast and the fact that it is an almost unavoidable consequence of the hypothesis provide strong support for the theory.

4. Felicity conditions: Further specification of the learning process

Not much as been said yet about the learning process, except that it is inductive and that it occurs in the context of a lesson sequence. Saying that learning is inductive is saying only that the input to the learning process is examples as opposed to, say, written recipes for performing the procedure. This section describes the particular kind of inductive learning that occurs in this domain.

Before beginning, it is important to establish the level of aggregation that will be employed. As Pylyshyn (1984), Newell and Simon (1972) and others have pointed out, it is important to characterize the behavior under study at a level of detail that is neither too fine, so that the important regularities are lost in a whelter of gratuitous

details, nor too gross, so that all the interesting behavior occurs "inside" the primitive components of the description. A practical difficulty has determined the level of aggregation employed in the present investigation, but the choice has proved a profitable one nonetheless. The difficulty is that learning is a long, complicated process in this domain. Consequently, the learning process must be described at a high level of detail. One way to indicate the level of aggregation of a process is to specify its "grain size," which corresponds to the smallest observable actions admitted under that level of analysis. For instance, in the earlier sections' description of test taking, the smallest observable action is writing a single digit. A finer-grained process would predict how the student writes a digit, i.e., the shape, sequence and timing of writing strokes. A larger grained process would predict, say, only the answers and not the sequence of writing actions used to produce them. Although the test-taking process can use digit-writing as its grain size, the learning process must be modelled at a much larger grain size. Students engage in so many different kinds of activities while learning procedures that a fine-grained model for all those activities would be inscrutably complicated or hopelessly incomplete. For instance, a process model that is detailed enough to account for the second-by-second learning behavior of a student being tutored would probably be inadequate to account for learning from textbook examples or from watching other students working problems at the blackboard. The variety of learning activities in this domain makes it mandatory that the theory employ a large-grained process to model skill acquisition.

The grain used in this theory corresponds, roughly speaking, to a single lesson*. One cycle of the learning model consist of taking in a lesson and a procedure, and producing a procedure. The procedures correspond to the students' procedural knowledge before and after the lesson. Actually, the model usually produces several post-lesson procedures. This amounts to the prediction that students may learn different things from the lesson, and so different students will acquire different procedures from the lesson even if they all started with the same pre-lesson procedure.

---

*Actually, by "lesson," I mean the introductory lesson for a topic and the drill lessons that accompany it. Often, these are grouped together as chapters or units in a textbook. They are quite clearly marked in the textbooks and the teachers' guides. I'll continue using the term "lesson" to refer to such collections of related activities. Also, I'm ignoring the spiralling structure of elementary mathematics curricula, where the previous year's lessons are reviewed before introducing this year's lessons.

Although the grain-size of the theory was set large for practical reasons, it was a providential choice. As will be shown in a moment, there are important regularities at this level of aggregation that have escaped the notice of educators and cognitive scientists, perhaps because they have viewed learning in too microscopic a way.

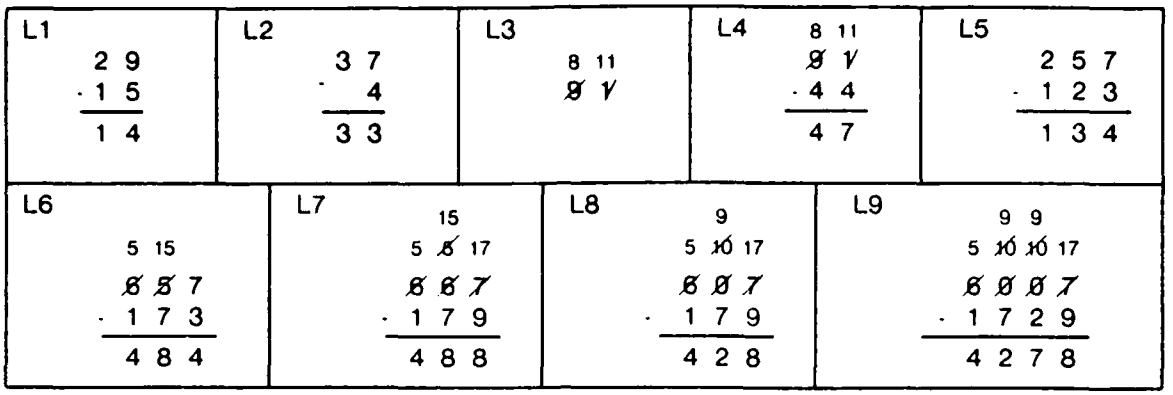
With the level of aggregation set, the central question can be addressed: what kind of inductive learning is taking place in this domain?

In principle, an inductive learning machine can be incredibly powerful (if it is given the right predilections for simplicity, representation, and so on). For instance, it would not be difficult to build an inductive learner that could learn all of subtraction from a single example, provided the example were long enough to display all the various subskills of subtraction, e.g.:

```
  4 9    4 1 1        5    4
  5 1 0 1 7  5  2 1 3  5  6 1 1  5 1 0  4  7  3  2  7
- 1 0 8 1 7  8  1 5 8  1  8 1 6  1 1 5
-------------------------------------------------------
  4 9 9 3 4  5  4 0 3  3  2 3 1  2 1 2
```

Donald Smith's learner (Smith, 1982) could probably handle this task with only a few modifications. However, children are not such powerful inductive learners. Their learning is some restricted form of induction. The job is to find out what those restrictions are.

One way to uncover the limitations on children's inductive power is to examine the difference between curricula that are learnable and those that are not. Before embarking on this comparison, it is important to clarify the learnability criterion. In this context, learnability is not meant to be a precise criterion. In particular, it is not meant to imply that all students finish the curriculum with a correct version of the skill. For instance, current mathematical curricula are learnable. Although not all students finish with a perfect understanding of the target skill, almost all students seem to learn something even if it is an incomplete or misconceived version of the skill. In contrast, the single-example curriculum mentioned above is unlearnable, because few students would learn anything from it, unless a teacher broke the example into parts and taught each part separately. But that would just convert the unlearnable single-example curriculum into a traditional, learnable, multi-example curriculum. Roughly put, a learnable curriculum is one from which almost all students can learn a general approximation of the target skill.



L1. Solving two columns
L2. Handling partial columns
L3. Regrouping
L4. Simple borrowing
L5. Solving 3 columns; no borrows
L6. One borrow in three columns
L7. Two adjacent borrows
L8 . Borrowing from one zero
L9. Borrowing from multiple zeros

Figure 3

A nine-lesson sequence. The topics of each lesson are listed in the lower part of the figure. Typical examples for each lesson are shown in the upper part.

Learnability is an objectively testible criterion, even though the learnability "facts" employed below are not the result of experimentation. Such experiments would be difficult and perhaps even immoral. The following discussion is intended to motivate and clarify certain hypotheses. It is not intended to be a convincing demonstration of their validity.

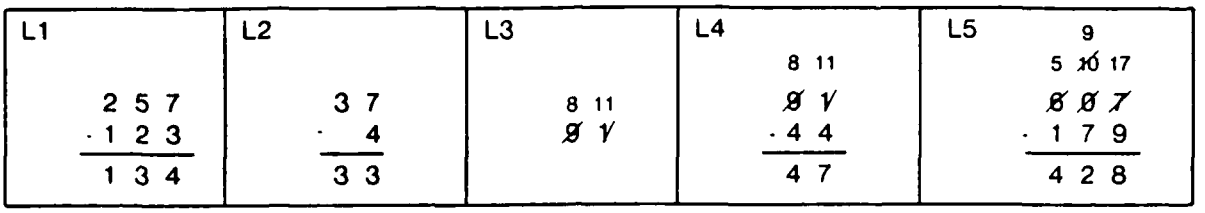
Figure 3 shows a subtraction curriculum from a popular American textbook series, published by Heath, and used by some of the schools we studied (VanLehn, 1983) It certainly qualifies as a learnable lesson sequence. Suppose one took all the examples used in this lesson sequence (there are probably thousands, if one counts the examples the teacher puts on the blackboard), randomized their order, and divided them into lessons of the same size as the original Heath lessons. This new curriculum would have exactly the same content and pacing as the Heath curriculum, but the examples would be in a different order, a random one. This curriculum would certainly be unlearnable. So, under one ordering, Heath's, the examples are learnable, but under another ordering, they are not learnable. Therefore, whatever the students' learning process is, it relies crucially on the ordering of the examples. This is an

important conclusion, for it eliminates a large class of potential hypotheses about the learning process, including most proposed models of natural language acquisition and of concept formation (see VanLehn, 1983, for a brief review, and Cohen & Feigenbaum, 1982, for a longer one), and most induction algorithms from recursive function theory (e.g., Gold, 1967; Angluin, 1980).

Imagine the Heath curriculum laid out as a long sequence of examples with marks that partition the sequences into lessons. Suppose one holds the sequential order of the examples the same, but moves the lesson boundaries around. For instance, a "lesson" in such a curriculum might have examples from Heath's L7 in its first half (i.e., two adjacent borrows: 542-168) and examples from L8 in its second half (i.e, borrowing across zero: 304-126). The only way for a teacher to make such a lesson sequence learnable would be to tell the students at the half-way point that a new subskill, borrowing across zero, is going to be introduced. This would, of course, convert the curriculum back into the Heath curriculum. If such shifted-lesson curriculum were taught straight, without the elaborations that would convert it back into the Heath curriculum, then it would be unlearnable. From this illustration, we can infer that learning depends crucially not only on the ordering of example, but on *how the examples are partitioned into lessons.*

Intuitively, the problem with the lesson that mixes L7 and L8 is that students will try to unify the ideas taught in the first half of the lesson with the ideas taught in the second half and end up with a confused mish-mash, because those two subskills have little in common. If this intuition is correct, then students can learn at most one subskill per lesson. Of course, the folklore of teaching endorses this by advising the teacher to teach slowly, one "topic" or "step" per lesson. If more than one topic must be taught during the alotted class time, then the teacher should divide the class time into mini-lessons, teach one concept per mini-lesson, and make it clear to the students where one mini-lesson ends and the next begins. Such advice about teaching is designed, I suggest, to accomodate a certain characteristic of students' learning processes, viz., that they learn at most one topic per lesson.

Because the material being learned in this domain is procedural, this key hypothesis will be rephrased as *students learn at most one subprocedure per lesson.* The question of what kind of learning occurs in this domain has been sharpened. Next we need a precise characterization of a subprocedure.



L1. Solving multiple columns
L2. Handling partial columns
L3. Regrouping
L4. Simple borrowing
L5. Borrowing from zeros

Figure 4

A very short lesson sequence.

One approach to finding a definition of subprocedure is to find a learnable curriculum that contains so much material per lesson that one can assume that the lesson "fills" the subprocedure to capacity. Such a lesson sequence would help one see the limits on what a subprocedure can be. Judging from a survey of textbook series, probably the shortest subtraction lesson sequence that is learnable is the one shown in figure 4.

Consider students who traverse this curriculum, and have the luck to make the right choices at every point where the lesson sequence is ambiguous. They will have correct, albeit incomplete procedures after each lesson. The new material added to their procedures will correspond to subprocedures. Figure 5 displays the appropriate procedures as augmented transition nets (ATNs). P0 is the assumed initial state of knowledge. The other P*i* correspond to the procedure after L*i*. The labels on the arcs stand for actions. Although arcs also bear conditions that say whether or not an arch should be traversed, those conditions are not shown in the figure.

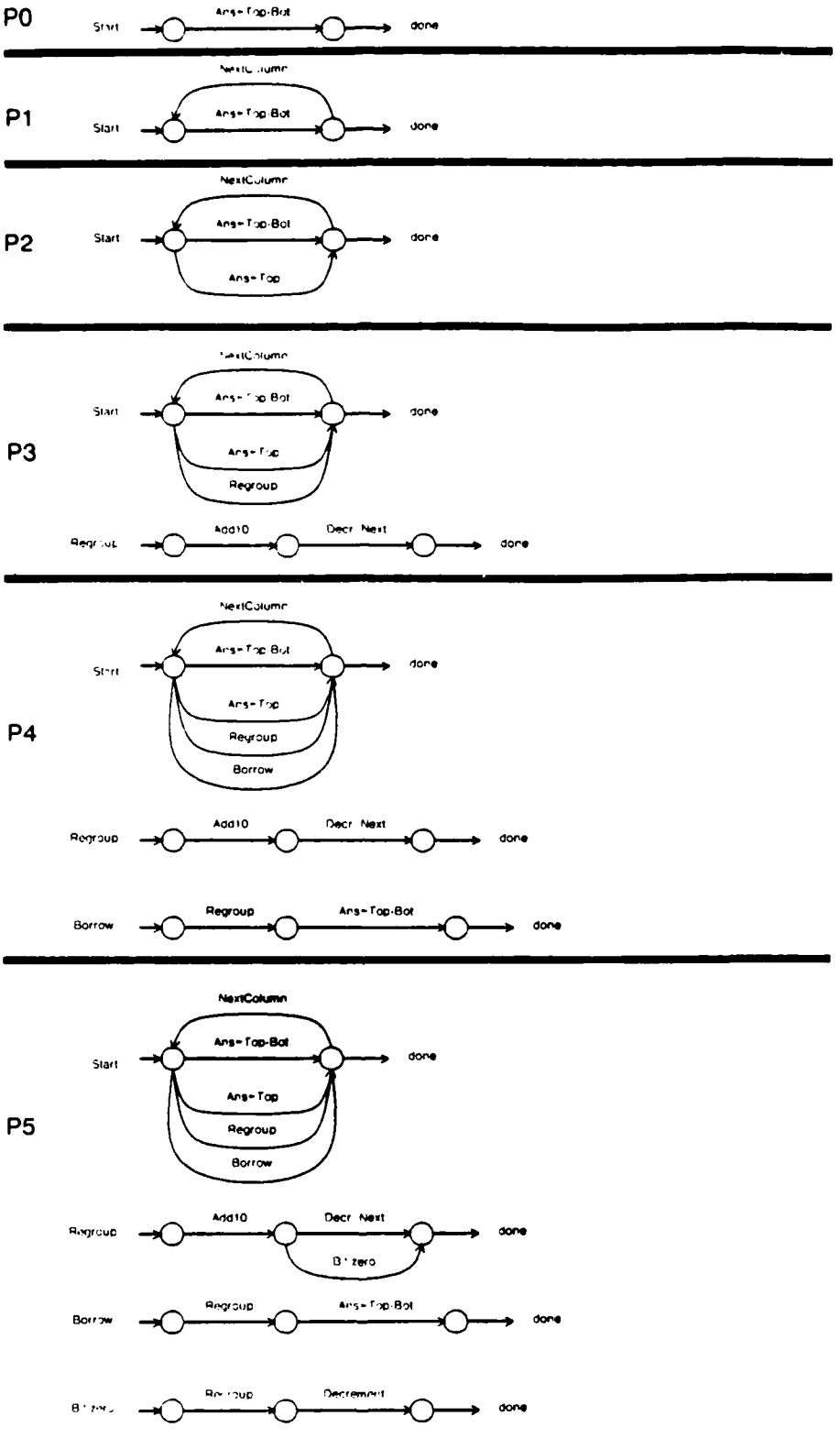


Figure 5
Procedures represented as ATNs

P0 is a procedure for solving single-column problems. It has a single non-trivial arc, labelled "Ans←Top–Bot," which stands for an action that subtracts the two digits in the column and writes their difference in the answer. (Top, Bot and Ans stand for the top, bottom and answer places in a column.) P1 is the procedure that results from taking lesson L1. The subprocedure added by L1 consists of an arc bearing the action NextColumn. This addition makes P1 able to iterate across columns, subtracting them. The subprocedure added by L2 is an arc to answer partial columns. The subprocedure added by L3 is an arc that calls Regroup and a new level to define the regrouping network. The subprocedure added by L4 is an arc that calls a new level, Borrow, that does borrowing from non-zero digits. L5 completes the procedure by adding a new level, labelled B.f.zero, that does borrowing from zero. In the ATN notation, it becomes quite clear that all the subprocedures share the characteristic that they add just one new "branch" or path to the procedure. In that notation, a subprocedure is an arc, plus an optional new level to define the action called by the arc, where the new level may not have branches.

This definition of "subprocedure" depends on notating procedures in a certain representation, ATNs. There doesn't seem to be any way around notating procedures in some way. However, the definition can be made more general and perspicous if procedures are notated in first order logic. This is similar in spirit to analyzing them at the knowledge level (Newell, 1982). Branches in the flow of control in a procedure become disjunctions (ORs) when the procedure is notated in first order logic. The one-ATN-arc-per-lesson constraint becomes one-*disjunct*-per-lesson when procedures are analyzed at the knowledge level.

The most direct way to test the one-disjunct-per-lesson hypothesis is to construct a curriculum whose lessons sometimes introduce more than one disjunct per lesson, then see if students can from it in their ordinary way. In some cases, such as merging lessons L4 and L5 in the above curriculum, I believe that the students will have a difficult time but they will manage to learn something. Would such a result refute the hypothesis? The answer depends on the ontological status one attributes to the hypothesis. I doubt that students are "hardware limited" in such a way that they simply cannot learn a lesson that introduces more than one disjunct. On the other hand, I doubt that they employ a uniform induction process that can gracefully learn subprocedures of any number of disjuncts, given enough time and willpower.

These beliefs follow in part from the fact that distinctly different induction algorithms are required for multi-disjunct-per-lesson learning than for one-disjunct-per-lesson learning, and that the multi-disjunct algorithms have much worse combinatorial properties. To use Brachman and Leveque's (1984) apt phrase, there is a *computational cliff* between one-disjunct-per-lesson learning and multi-disjunct-per-lesson learning. The existence of such a cliff is well known. There are a variety of formal results that show that induction with disjunctions is hard or even impossible, while induction of disjunction-free concepts can be achieved quite economically (Berwick, 1983; Angluin, 1980). One such result is particularly interesting, because it refers to concepts expressed in first-order logic, which is the notation used for knowledge level analysis. It can be shown (VanLehn, forthcoming-b) that there are exactly three constructions in first order logic that cause computational cliffs (put more technically, they cause the number of expressions consistent with any finite set of examples to become infinite):

1. Disjunction
2. Function nesting (e.g., f(g(x)) where f and g are functions)
3. Quantifier scoping (e.g., For all x, there exists a y ...)

Suppose that computational cliffs cause the teacher-student cultural system to evolve conventions that help the student climb the cliff, so to speak. The conventions dictate that the teacher gives the student certain kinds of hints whenever the students is faced with a computationally intractible induction task. The convention not only leads the student to expect such hints, but more importantly, it tells the student how to interpret them. Because the hints, under the interpretation of the convention, provide extra information beyond the mere examples, the student can employ modified, quasi-inductive learning processes that can acquire the troublesome constructions. These learning process remain tractible because they utilize extra information that pure induction does not. For reasons that will be discussed shortly, this conjecture will be called the *felicity conditions conjecture*, and the conventions that have evolved to facilitate learning will be called *felicity conditions*.

If the felicity conditions conjecture is right, then there should be felicity conditions for disjunctions, function nesting and quantifier scoping, as these are the constructions that cause computational cliffs. One-disjunct-per-lesson is the felicity condition for disjunction. What about the other two?

The cliff caused by function nesting is due to the fact that when functions are nested, the intermediate results are not constituents of the examples. Without being

able to see the input-output relations of each function in the nest, it is difficult to induce them. To put it intuitively, if you are given few number triples, such as [1,2,2] and [5,1,5], and asked to induce what the numerical relationship among the numbers is, then the task is trivial when you are guaranteed that expressing the relationship does not require nesting arithmetic functions. For the triples just given, the only answer is $x*y=z$ (and its logical equivalents, of course). However, if nesting is allowed, then the answer could be $x^2+y^2=z^2+1$. If you could see all the intermediate results, namely x^2, y^2, x^2+y^2, and z^2 in the latter case, and you were informed that *all* the intermediate results were listed in the example tuples, then the problem would once again be trivial.

Is there any evidence of a felicity condition for function nesting in mathematical curricula? As it turns out, subtraction does not employ any hidden, intermediate results. Scratch marks are used instead. However, adding three or more numbers does require hidden intermediate results. If it were learned by induction, then the learner would have to climb a computation cliff in order to discover the appropriate nesting of functions. Multi-addend addition is an appropriate place to look for a felicity condition concerning function nesting.

Textbooks usually teach three-number addition in two adjacent lessons. The first lesson uses an ad hoc notational format that provides a place for the intermediate result to be written down. Figure 6 shows some of the formats used. Because the intermediate results are made visible in the examples, the students can induce a three-number procedure without climbing the computational cliff. The next lesson is specially marked. In fact, most textbooks title the lesson "A shorter way to add." Such labels, plus the teacher's explanations of course, inform the learner that this lesson will not be an induction lesson. Rather, the same old stuff is going to be accomplished in a new way by suppressing some of the writing. That is, they are going to hide a previously visible intermediate result by creating a nest of two functions that are already present in the procedure. The combinatorial problems involved in doing this are trivial. An impossibily difficult induction problem has been converted into two simple problems by adopting a convention. Normal lessons always "show all the work." Only specially marked "hide work" lessons introduce function nests. This felicity condition is called the show-work convention. If it is an accurate characterization of the teacher-student system, then special formats and special hide-work lessons should be found whenever a procedure employs a hidden intermediate result. Of 14 cases over 6 textbooks, there were only 4 violations of the show-work condition. Figure 7 shows some illustratory formats.

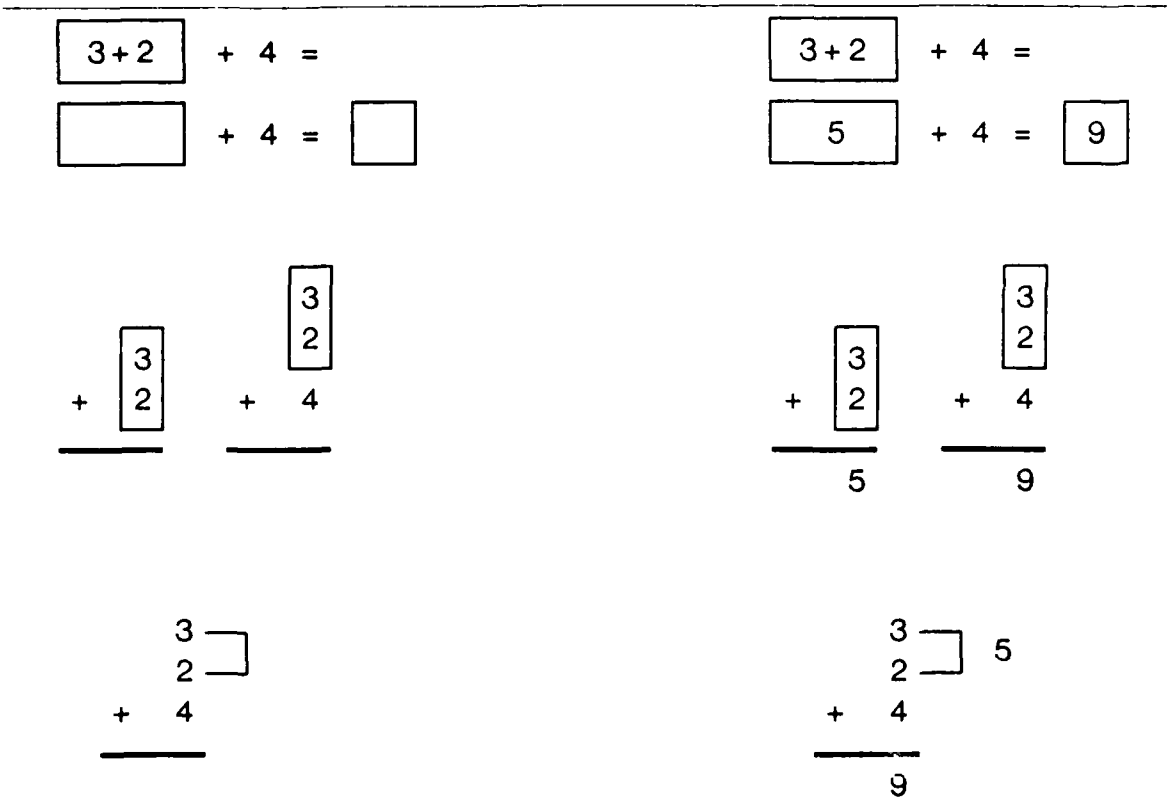


Figure 6

Show-work formats for multi-row addition.
Problems are shown unsolved on the left and solved on the right.

Is there a felicity condition for quantifier scoping? Quantifier scoping is rarely used in mathematical procedures. I've only found one case, namely the definition of the concept of "like terms," which is employed in high school algebra. Textbooks use a special lesson for this concept that includes negative examples. (Negative examples are cases that the target concept should *not* match, whereas normal examples—also called positive examples—are cases that the target concept *should* match.) Such lessons are the only ones I know of that employ negative examples. There is probably a felicity condition involved, but it is best to collect more cases before attempting to state it precisely.

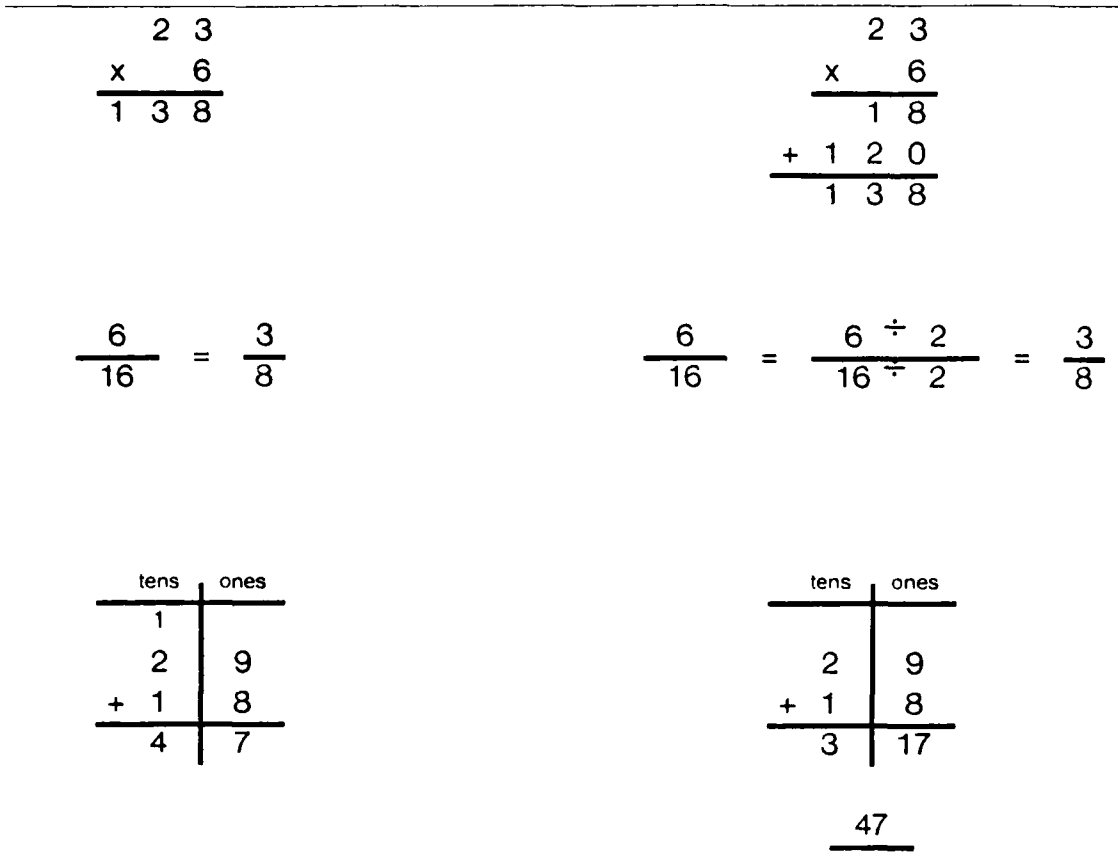


Figure 7

Regular formats on the left; show-work formats on the right.

In short, it seems that the felicity-condition conjecture holds for mathematical skill acquisition. For each computational cliff, there is a felicity condition. This remarkable three-way convergence of evidence is a major piece of support for the felicity-condition conjecture.

Some informal evidence of the existence of felicity conditions has been presented. But why should they exist? What would explain their existence? There are several background assumptions that, taken together, imply that it is no accident that felicity conditions for skill acquisition exist. They also give felicity conditions their odd name.

It is quite common, especially in AI, to assume that skill acquisition consists of a combination of communication and compilation. The teacher somehow communciates a skill to the students, and the students somehow compile their received understanding into a smoothly operative form as they practice. To put it differently, learning = communication + compilation. There have been many studies of practice-driven compilation (e.g., Anderson, 1983; Rosenbloom & Newell, 1981). Complementary to those, this study concentrates on the communication half of the equation.

Assuming skill acquisition involves a form of teacher-student communication, then it ought to be like other forms of human communication in that the participants followed certain conventions that make the communication smoother and more reliable. Such conventions have been extensively studied in natural language conversations, where they are often called felicity conditions (Austin, 1962), or conversational postulates (Gordon & Lakoff, 1971), or conversational maximes (Grice, 1975). A typical linguistic felicity conditions is: In normal conversation, the speaker uses a definite noun phrase only if the speaker believes that the listener can unambiguously determine the noun phrase's referent (Clark & Marshall, 1981). Typically, neither the speaker nor the hearer is aware of such constraints. Yet if a conversation violates a felicity condition, it is somehow marked, e.g., by the speaker appearing sarcastic or the hearer misunderstanding the speaker. Although felicity conditions for conversations probably are not identical to felicity conditions for skill acquisition, it is apparent that they share the secondary characteristic that the participants in the communication are not aware of the rules they are following. Teachers and textbook authors probably do not consciously realize that the lessons they write obey, e.g., the one-disjunct-per-lesson constraint. They strive only to make the lessons effective. The students do not realize that the "obvious" interpretation of the lesson is the one-disjunct-per-lesson interpretation.

Part of the reason for believing that natural language is governed by conventions is that humans have been talking to each other for so long that cultural evolution, or perhaps even biological evolution has had ample time to develop constraints that make tend to make communication more efficient. The same (weak) reasoning applies to teacher-student communication, for humans were probably teaching other humans how to do things long before they began talking to each other. Our culture/species has had sufficient opportunity to evolve conventions on how to teach and how to learn. Perhaps efficient customs for teaching/learning impart a survival advantage to a species.

5. Methodology

Most of the important claims of the theory have been presented. To summarize them briefly, they are:

- When students reach an impasse while solving a test exercise, they repair. Repairs change the state of the interpretation, but not the test exercise or the procedure.
- Students induce at most one subprocedure per lesson, where the definition of "subprocedure" embodies the one-disjunct-per-lesson hypothesis, the show-work hypothesis, and several other hypotheses.
- One-disjunct-per-lesson and the other constraints on induction are probably deeply ingrained cultural conventions, and are thus called felicity conditions.

The first two are bona fide hypotheses of the theory, while the third is a conjecture that would take a completely different sort of theory to test. The hypotheses have been illustrated with bugs and other empirical material. However, these facts were offered only as a way to explain the hypotheses, and not as validation for them. This section describes the validation method.

Logically, all one needs to validate a theory is a formal statement of the theory's hypotheses, a way of deriving the empirical entailments of the hypotheses, and a way of testing those predictions. The validation is made a bit more elaborate in this case because the theory has a large number of hypotheses. There are 31 major hypotheses, and several more minor ones. Even if the best theorem provers were used, it would probably not be possible to generate predictions directly from the hypotheses. An intermediate stage is used. A computer program, Sierra, has been built to instantiate the hypotheses in a form that can efficiently generate predictions (see VanLehn, 1985a, for a description of Sierra). Sierra simulates the learning and problem solving processes hypothesized by the theory. When given (1) a formalized version of the lesson sequence taken by some students and (2) a formalized version of the diagnostic test taken by the students, Sierra predicts what the students' bugs will be.

Logically, it is necessary to *prove* that Sierra computes the same input-output function as the hypotheses. This is a familiar, but difficult task in computer science: given a set of formal specifications and a program, verify that the program meets the specifications. Although the technology of program verification is improving steadily, it is not sufficiently developed that a formal verification of Sierra can be written.

Informal techiques have been used. For instance, whenever possible, the program simple generate-and-test algorithms where the tests correspond to hypotheses of the theory. This slows the program down, but makes it easier to see that it is generating what the hypotheses say it should be generating.

The major empirical test of the theory is its ability to predict the occurrence and non-occurrence of bugs. More specifically, Sierra generates a set of predicted bugs, and the students produce a set of observed bugs. Ideally, every observed bug is a predicted bug, but not vice-versa. Even with an ideal theory, one would not want every predicted bug to be an observed bug. Because only a finite sample of the world's students have been tested, one would not expect every possible observed bug to show up in the sample. So the theory should predict some bugs that haven't yet been observed. To put it more formally, if P is the set of predicted bugs and O is the set of observed bugs, then

1. $O \cap P$ should be large,
2. O–P should be small, and
3. P–O should be non-empty. It is the prediction of future observations.

These criteria have a loophole. Consider a trivial theory that generates a huge set of predictions, e.g., all logically possible procedures. Thus, $O \subset P$ and the trivial theory meets all three criteria. However, the theory is clearly empirically inadequate because it predicts bugs that probably would never be observed. It should be penalized for such overgeneration. However, this requires somehow deciding which of its predictions would never occur, and that is necessarily a non-objective judgment. All generative theories have this same problem. In generative theories of grammar, the custom is to call ungrammatical sentences "star" sentences, and label them with an asterick. In the judgment of native speakers, such sentences would never occur in written or spoken language. That custom has been adopted here. *Star bugs* are bugs that are so implausible that, in the opinion of experienced teachers and diagnosticians, the bug will never occur. Often, it is quite clear when a bug is a star bug. For instance, Sierra has occasionally generated a procedure that performs all types of borrowing with perfect competence, yet it leaves the answer to the problem blank. This unlikely juxtaposition of competence and incompetence makes this behavior a star bug.

As an illustration of this kind of empirical testing, figure 8 shows the bug counts for a 1147-student sample of subtraction students. The students (and Sierra) used the Heath subtraction curriculum (Dilley, Rucker & Jackson, 1975) and a similar

curriculum from Scott-Foresman (Bolster et al., 1975). There were 79 observed bugs, of which 22 (28%) were predicted by the theory. The remaining 57 bugs should have been predicted by the theory. However, they are not a serious problem for several reasons. First, some of the 57 bugs could be generated by the theory if the model were given lesson sequences other than ones from Heath and Scott-Foresman. It is indeed plausible that some students transferred into our sample schools from schools that use different textbook series. However, we did not have access to the students' educational history and therefore could not ascertain all the lesson sequences that the students may have learned from. We have conservatively chosen to evaluate the model using only lesson sequences that we are certain were administered to the subjects.

There is a second reason why the predicting only 28% of the observed bugs is not a serious problem. It is simple to relax the theory's hypotheses in such a way that significantly more observed bugs are predicted. Under one relaxation (Brown & VanLehn, 1980), 48% of the observed bugs are predicted. Under another (VanLehn, 1985b), 85% are predicted. However, such relaxations also cause the theory to generate more star bugs. Currently the theory predicts just three star bugs. The hypotheses have been adjusted to reduce the number of star bugs to a minimum. The reason for taking this stand is that more than one learning process may be going on in this domain, and some of the 57 bugs could come from those other processes. However, augmenting the theory with another learning process will not block the generation of the three star bugs.

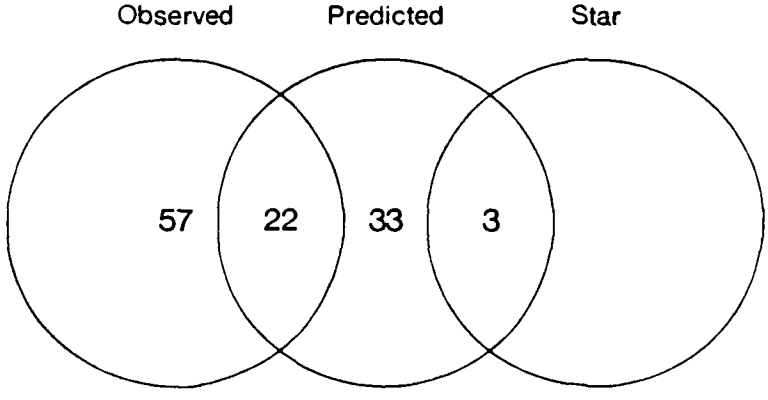


Figure 8
Observational adequacy circa March, 1983

It should be obvious that merely counting the observed bugs and star bugs generated by the theory is not a revealing measure of its empirical adequacy, particularly when one can "trade" observed bugs for star bugs and vice-versa. It is impossible to tell from the bug counts whether the theory is terribly wrong or only slightly askew. Worse, one can not tell whether *individual hypotheses* are right or wrong. One can't tell, for instance, whether one-disjunct-per-lesson is an accurate characterization of students' learning. The overall observational adequacy of the theory just doesn't suffice to answer the really interesting questions.

Ideally, we would perform a perturbation analysis of the theory. As there are 31 major hypotheses, and we want to assign empirical credit or blame to each hypothesis individually, suppose we form 31 new sets, each with a single hypothesis deleted (i.e., 31 sets of 30 hypotheses each), then revise Sierra appropriately and generate 31 new sets of predictions and their corresponding bug counts. Better still, alternatives to the various hypotheses would be substituted into the set of 31, and the resulting observational adequacy would be measured. Such an analysis would allow us to assess the empirical responsibility of each hypothesis and contrast its performance to competing, alternative hypotheses.

Unfortunately, the hypotheses of the theory are not independent. In general, one can't just remove a hypothesis or substitute an alternative hypothesis without also modifying several other hyptheses in order to accomodate the change. This does not make a perturbation analysis impossible, but it does make it more complicated.

A technique has evolved for doing such analyses. VanLehn, Brown and Greeno (1984) call it competitive argumentation, and show how it can solve many of the methodological problems that plague current cognitive science. Most competitive arguments have a certain "king of the mountain" form. The argument shows that a hypothesesis accounts for certain facts, and that certain alternative hypotheses, while perhaps not without empirical merit, are flawed in some way. That is, the argument shows that its hypothesis stands at the top of a mountain of evidence, then proceeds to knock the competitors down. Perhaps the best way to describe competitive arguments is to present an example of one.

Earlier, an explanation for the bug Always-Borrow-Left was presented. Part of the explanation involved how students acquired a description of which column to borrow from. The claim was that, given two-column problems as examples, students would induce that the column to borrow-from is *both* the leftmost column in the problem and the column that is left-adjacent to the column causing the borrow.

Because both descriptors are included, the student reaches an impasse on three-column borrowing problems. For the problem

```
  6 2 4
- 1 5 7
-------
```

the hundreds column is the leftmost column and the tens column is the left-adjacent column. On this problem, the induced description is unsatisfiable. This causes an impasse. One repair, ignoring the left-adjacency descriptor, generates Always-Borrow-Left. Another repair, skipping the decrement entirely, generates Borrow-No-Decrement-Except-Last. A third repair, ignoring the left-most descriptor, generates a correct subtraction procedure.

However, why should a student include *both* descriptors in the description? A plausible alternative hypothesis is that students only include enough material in the borrow-from column's description to differentiate that column from the others. This kind of induction is called *discrimination*. The other kind of induction, which puts both descriptors in the borrow-from column's description, is called *generalization* (Langley et al., 1980). So there are two competing hypotheses. As just mentioned, the generalization hypothesis generates three bugs, all of which occur. It also predicts, correctly, that there will be bug migrations among the three bugs. Let's see what the discrimination hypothesis predicts.

Under the discrimination hypothesis, some students will induce left-most as the discriminating feature of the borrow-from column, and other students will induce left-adjacent as the discriminating feature (of course, there could be other discriminating features, but just two will be used to keep the illustration simple). A student who thinks the column to borrow-from is the left-most column will have the bug Always-Borrow-Left. A student who thinks the borrow-from column is left-adjacent to the borrow-into column will have a correct subtraction procedure. Neither student will reach an impasse. Their descriptions are always satisfiable and unambigouous. Consequently, there is no way to generate the third bug, Borrow-No-Decrement-Except-Last, which skips the borrow-from action in certain circumstances. Moreover, there is no way to generate a bug migration. So the discrimination hypothesis can only generate two of the four predictions that the generalization hypothesis makes.

Moreover, the predictions that the discrimination hypothesis misses are not generated by other mechanisms in the model. The only known derivation of

Borrow-No-Decrement-Except-Last and the bug migration is the deriviation that requires the generalization hypothesis. Thus, some students must be employing generalization. Other students could be employing either generalization or discrimination. In order to totally defeat the discrimination hypothesis, it would be necessary to show that discrimination generates *only* star bugs. This cannot be done. It has already been shown that discrimination generates some observed bugs.

However, there is weaker argument against discrimination. It is based on parsimony. Generalization alone covers all the data. Discrimination alone does not cover all the data, so that hypothesis is out. However, discrimination is consistent with some of the data, so it might be that some students generalize and other discriminate. The generalization-plus-discrimination hypothesis adds no additional coverage when compared to the generalization hypothesis, but it does add an additional mechanism (and a parameter of between-subjects variability, which raises the issue of explaining why some students generalize and other discriminate). By Occam's razor, the simpler, one-mechanism hypothesis is preferred. The generalization hypothesis wins the competitive argument.

With this illustratory argument in hand, several methodological points can be made. First, there is no a priori source of competing hypotheses. In this case, the hypotheses concern concept formation, on which there is a large literature containing many hypotheses (see Anderson, Kline & Beasley, 1979, for a comparative review). Discrimination and generalization are perhaps the two most important hypotheses. A full-fledged competitive argument would contrast all the hypotheses mentioned in the literature. Of course, there are infinitely many hypotheses that haven't yet appeared in the literature. Logically, the argument is incomplete until they too have been included. But this is just the normal incompleteness of empirical science. The best one can ever do is to show that the present hypotheses is the best of the known hypotheses. Later, a better hypothesis might be invented. Indeed, the hope is that one will be discovered.

It was announced at the outset that one goal of the present research is a parameter-free learning theory. The preceding argument indicated that the theory has at least one parameter, namely, the vocabulary of visual features, such as left-most and left-adjacent, that are used to build descriptions of locations in exercise problems. These visual features are primitives, in that their definitions are not constructed by the learning model. Instead, the theorist chooses which primitives to employ, then writes the code that defines them. Besides primitives for visual descriptions, the theory requires primitives for representing writing actions and arithmetic facts (e.g., both "<" and "$\leq$" are provided as arithmetic primitives). The set of primitives is the theory's

only parameter*. So the theory turns out to be a one-parameter theory. It is hard to imagine a cognitive model that doesn't have primitives of some kind. I suspect, therefore, that all cognitive theories will have at least one parameter, the set of primitives they employ. So a zero-parameter theory is probably impossible. However, one can try to arrange the theory so that it is relatively insensitive to the choice of primitives. Indeed, several competitive arguments are decided by the desire to curtail the sensitivity of the theory's predictions to its parameter values.

Arguments generally need to take certain hypotheses as givens. For the argument given above, it was assumed (1) that learning was inductive, (2) that impasses occur when descriptions are unsatisfied or ambigouous, and (3) that impasses are repaired in certain ways. It is important to check for circularities in these assumptions. It would be incorrect for the argument supporting hypothesis A to assume hypothesis B, and the argument for B to assume A. Moreover, whenever new evidence is discovered that changes the conclusion of an argument, all the arguments that depend on its conclusion have to be reexamined to see if they still go through. Maintenance of the argumentation structure became such a significant problem that a computer program, the Xerox Notecards system, was enlisted to help. VanLehn (1985c) describes the system and how it was used.

If most hypotheses depend on other hypotheses for their support, then there must be a few hypotheses that don't depend on any other hypotheses. It is almost impossible to justify such hypotheses in any rigorous way. Such hypotheses are called assumptions. The induction hypothesis is one. It, and the other assumptions, are supported by informal observations and, indirectly, by the sucess of the theory as a whole.

The logical structure of the theory is an acyclic directed graph (i.e., tangled hierarchy, partial order). The nodes are the competitive arguments. Node A depends on node B if the A argument takes B's conclusion as a given. The assumptions are nodes that do not depend on any other nodes. This logical structure is a familiar one to AI researchers. It is sometimes called a data dependency net or a dependency graph. A truth maintenance system (de Kleer's ATMS, in press) may be useful in helping manage the argumentation, with the theorist playing the role usually performed by a theorem prover or other problem solver. Such sophisticated support for theory development may be extremely valuable in coping with the complexity of validating a theory of this size.

---

* Actually, there is a second parameter, a grammar for the syntax of the exercise problems. It is used to define primitives like "column." See VanLehn (1983) for details.

Cognitive theories of skill acquisition, and cognitive science in general, are entering a new phase in which such technology for supporting theorists will become increasingly important. In the early years of AI, it was considered ar impressive accomplishment to get a machine to learn a simple skill, such as recognizing towers and arches made of toy blocks. The mere fact that the machine could perform the skill acquisition was taken as an argument that the processes it employed were plausibly the ones that humans used. Standards escalated in later years. As protocol analysis became common, psychological claims about cognitive processes had to be accompanied by a detailed match between the machine's performance and the subject's performance. Learning theories temporarily disappeared during this era, because protocols of subjects learning complex skills are intractibly long. However, these two earlier eras were of fundamental importance, even though they did not yield scientifically adequate theories of skill acquisition, because they developed the computational and methodological tools that the present era uses.

Nowadays, theories of skill acquisition are reappearing. There are at least three theories presently under developement: ACT (Anderson, 1983), SOAR (Laird,1983; Rosenbloom, 1983; Laird, Rosenbloom & Newell, 1985) and the theory described here. Competitive argumentation will become increasingly important in order to to compare these theories. Indeed, competitive argumentation seems necessary just to cut through the jargon and see whether or not two theories are actually the same. To put it bluntly, after several decades of struggling to understand the computational medium, computational research on skill acquisition has finally arrived at a place where it can begin to embrace rigorous scientific methods. It is fortunate that the technology for supporting the complicated competitive argumentation is here, because we need it to do science.

7. Conclusions

Much material has been covered in a brief, intuitive way. The main points, however, can be simply summarized. Three major claims have been made about human cognition:

- When students reach an impasse while solving a test exercise, they repair. Repairs change the state of the interpretation, but not the test exercise or the procedure.
- Students induce at most one subprocedure per lesson, where "subprocedure" is defined by one-disjunct-per-lesson, show-work and several other hypotheses.

- One-disjunct-per-lesson and the other constraints on induction probably are deeply ingrained cultural conventions, and are thus called felicity conditions.

There are other important claims made by the theory beyond the three listed above. Some of the most important concern the structure of procedures. For instance, one hypothesis is that procedures are hierarchical. A procedure is a tree of goals, and its interpretation requires maintaining a goal stack. This hypothesis is similar to ones in ACT (Anderson, 1983) and SOAR (Laird, Rosenbloom & Newell, 1985), which also postulate goal hierarchies as the organization of problem solving knowledge. Another hypothesis, which is not shared by those theories, is that focus of attention is maintained applicatively (VanLehn, 1983). That is, goals are instantiated with a focus of attention directed at a specific region of the problem state. The focus may not be changed during the (brief) lifetime of the goal instance. In ACT, for instance, focus is maintained in a global resource, the working memory, which may be changed arbitrarily during the processing of goals. To put it crudely, ACT uses Fortran-like global variables to hold focus of attention, while this theory uses lambda-calculus-like local variables to hold focus of attention.

Such hypotheses about the mental representation of procedures have far reaching consequences. They effect both the kinds of learning processes that can be employed to acquire them, and the kinds of problem solving processes that can be employed to interpret them and solve exercise problems. The ontological status of the putative mental representations is a matter of some interest (Fodor, 1975). In what sense is mental information stuctured? What does it mean for procedures to be tree-structured and applicative? These are very subtle issues, and a short paper like this one can not do them justice. Neither can it review, unfortunately, the evidence that supports the theory's specific claims about mental representations, for the arguments that support those hypotheses are some of the most intricate in the whole theory.

Roughly half of this article has discussed methodology: the challenges of validating a complicated theory of cognitive processes, and the techniques that have evolved to meet those challenges. To summarize briefly, the main methodological points are:

- Program parameters should be eliminated from cognitive theories. This has recently become feasible as the computer power required for learning theories is now available. This theory has no program parameter. Its only parameter is a set of primitives used to represent problem states, actions and arithmetic facts.

- The theory has a well-defined empirical test: it should generate all the observed bug data without generating implausible predictions (e.g., star bugs).
- The theory has both a set of formal hypotheses and a computer program. The program is claimed to be logically equivalent to the hypotheses, but more efficient for generating the theory's predictions.
- Each hypothesis has been individually motivated by examining the empirical entailments of alternatives to it. Such competitive argumentation is vital for complex cognitive theories but difficult to accomplish. Fortunately, argumentation support tools, which have recently become available, make the job easier.

In many ways, the stage is set for rapid advances in cognitive science. The decades spent in coming to grips with the computational medium have paid off in a technology suitable for modelling cognitive processes, especially learning. Intelligent tutoring and diagnostic systems (Sleeman & Brown, 1982) make it simple to collect and analyze detailed data from thousands of students. The refined methods of linguistic inquiry have been successfully adapted for supporting generative theories in non-linguistic domains. The evolution of text editors and database tools into "idea editors" provide an appropriate medium for developing the argumentation structures needed to connect complicated cognitive simulation programs to rich behavioral data.

7. References

Angluin, D. (1980) Inductive inference of formal languages from positive data. *Information and control*, 45, 117-135.

Anderson, J.R. (1983) *The Architecture of Cognition*. Cambridge, MA: Harvard.

Anderson, J.R., Kline, P.J. & Beasley, C.M. (1979) A general learning theory and its application ot schema abstraction. *The Psychology of Learning and Motivation*, 13, 277-318.

Ashlock, R.B. (1976) *Error patterns in computation*. Columbus, OH: Bell and Howell.

Austin, J.L. (1962) *How to do things with words*. New York: Oxford University Press.

Berwick, R.C. (1983) Domain-specific learning and the subset principle. *Proceedings of the International Machine Learning Workshop*. University of Illinois, Urbana, IL, 228-233.

Bolster, L.C., Cox, G.F., Gibb, E.G., hansen, V.P., Kirkpatrick, J.E., Robitaille, E.F., Trimble, H.E., Vance, I.E., Walch, R. & Wisner, R.J. (1975) *Mathematics Around Us*. Glenview, IL: Scott-Foresman.

Brachman, R. & Leveque, H. (1984) The tractability of subsumption in frame-based description languages. *Proceedings of the National Conference on Artificial Intelligence*. Los Altos, CA: Kaufman.

Brown, J.S. & Burton, R.B. (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. (1941) The evaluation of learning in arithmetic. In *Arithmetic in general education*. 16th yearbook of the national Council of Teachers of Mathematics. Washington, DC: NCTM.

Brueckner, L.J. (1930) *Diagnostic and remedial teaching in arithmetic*. Philadelphia, PA: Winston.

Burton, R.B. (1982) Debuggy: Diagnosis of errors inbasic mathematical skills. In D. H. Sleeman & J.S. Brown (Eds.) *Intelligent Tutoring Systems*. New York: Academic.

Buswell, G.T. (1926) *Diagnostic studies in arithmetic*. Chicago, IL: University of Chicago Press.

Clark, H.H. & Marshall, C.R. (1981) Definite reference and mutual knowledge. In A. K. Joshi, B.L.Webber & I.A.Sag (Eds.) *Elements of discourse understanding*. New York, NY: Cambridge University Press.

Cohen, P.R. & Feigenbaum, E.A. (1983) *The Handbook of Artificial Intelligence, Vol. 3*. Los Altos, CA: William Kaufmann.

Collins, A.M. & Loftus, E.F. (1975) A spreading activation theory of semantic processing. *Psychological Review*, 82, 407-428.

Cox, L.S. (1975) Diagnosing and remediationg systematic errors in addition and subtraction computation. *The Arithmetic Teacher*, 22, 151-157.

de Kleer, J. (in press) An assumption-based truth maintenance system. *Artificial Intelligence*.

Dilley, C.A., Rucker, W.E. & Jackson, A.F. (1975) *Heath Elementary Mathematics*. Lexington, MA: Heath.

Durnin, J.H. & Scandura, J.M. (1977) Algorithmic approch to assessing behavior potential: comparison with item forms. In J. M. Scandura (Ed.) *Problem Solving: a structural/process approach with instructional implications*. New York: Academic.

Fodor, J.A. (1975) *The language of thought*. New York: Crowell.

Gold, E.M. (1967) Language identification in the limit. *Information and Control, 10*, 447–474.

Gordon, D. & Lakoff, F. (1971) Conversational postulates. In D. Adams, M. A. Campbell, V. Cohen, J. Lovins, E. Maxwell, C.Nygeren, & J. Reighard (Eds.) *Papers from the seventh regional meeting of the Chicago Linguistic Society*. Chicago: University of Chicago Department of Linguisites, 63-84.

Grice, H.P. (1975) Logic and conversation. In D. Davidson & G. Harmon (Eds.) *Semantics of Natural Language.* Dordrecht: Reidel Press.

Laird, J. (1983) *Universal Subgoaling.* (Tech. Rep. 84-129) Pittsburgh, PA: Carnegie-Mellon University Computer Science Dept.

Laird, J., Rosenbloom, P. & Newell, A. (in prep.) The generality of a simple learning mechanism: Chunking in SOAR.

Lankford, F.G. (1972) *Some computational straategies of seventh grade pupils.* (ERIC document) Charlottesville, VA: University of Virginia.

Langley, P., Neches, R., Neves, D., & Anzai, Y. (1980) A domain-independent framework for procedure learning. *Policy Analysis and Information Systems,* 4, 163-197.

Newell, A. (1973) Production systems: Models of control structures. In W. C. Chase (Ed.) *Visual Information Processing.* New York: Academic.

Newell, A. (1978) Harpy, production systems and human cognition. In R. Cole (Ed.) *Perception and Produciton of Fluent Speech,* New York: Erlbaum.

Newell, A. (1980) Reasoning, problem solving and decision processes: The problem space as a fundamental category. In R. Nickerson (ed.) *Attention and Performance VIII.* Hillsdale, NJ: Erlbaum.

Newell, A. (1982) The Knowledge Level. *Artificial Intelligence,* 18, 87-127.

Newell, A. & Simon, H.A. (1972) *Human Problem Solving.* Englewood Cliffs, NJ: Prentice-Hall.

Norman, D.A. (1981) Categorization of Action Slips. *Psychological Review,* 88, 1-15.

Pylyshyn, Z. W. (1984) *Computation and Cognition.* Cambridge, MA: Bradford Books.

Roberts, G.H. (1968) The failure strategies of third grade arithmetic pupils. *The Arithmetic Teacher,* 15, 442-446.

Rosenbloom, P.S. (1983) The Chunking of Goal Hierarchies: A Model of Practice and Stimulus-Response Compatibility. Unpublished Ph.D. dissertation. Carnegie-Mellon University.

Rosenbloom, P. & Newell, A. (1981) Mechanisms of skill acquistion and the law of practice. In J.R. Anderson (Ed.) *Cognitive Skills and their Acquisition.* New York: Erlbaum.

Shaw, D.J., Standiford, S.N., Klein, M.F. & Tatsuoka, K.K. (1982) Error analysis of fraction arithmetic — selected case studies (Tech. Rep. 82-2-NIE) Urbana, IL: University of Illinois, Computer-based Education Research Laboratory

Siegler, R.S. & Shrager, J. (in press) Strategy choices in addition: How do children know what to do? In C. Sophian (Ed.) *Origins of cognitive skill.*

Sleeman, D.H (1984) Basic algebra revised: A study with 14-year olds. *International Journal of Man-Machine Studies.*

Sleeman, D.H. & Brown, J.S. (1982) *Intelligent Tutoring Systems.* New York: Academic.

Smith, D.E. (1982) Focuser: A strategic interaction paradigm for language acquisition (Tech. Rep. LCSR-TR-36). Rutgers, NJ: Laboratory for Computer Science Research, Rutgers University.

Tatsuoka, K.K. & Baillie, R. (1982) Rule space, the product space of two score components in signed-number subtraction: an approach to dealing with inconsistent use of erroneous rules (Tech. Rep. 82-3-ONR) Urbana, IL: University of Illinois, Computer-based Education Research Laboratory.

VanLehn, K. (1982) Bugs are not enough: Empirical studies of bugs, impasses and repairs inprocedural skills. *The Journal of Mathematical Behavior*, 3, 2, 3-71.

VanLehn, K. (1983) Felicity conditions for human skill acquisition: Validating an AI-based theory (Tech. Rep. CIS-21) Xerox Palo Alto Research Center, Palo Alto, CA.

VanLehn, K. (1985a) Learning one subprocedure per lesson (Tech. Rep. ISL-10). Palo Alto, CA: Xerox Parc.

VanLehn, K. (1985b) Arithmetic procedures are induced from examples (Tech. Rep. ISL-12). Palo Alto, CA: Xerox Parc.

VanLehn, K. (1985c) Theory reform caused by an argumentation tool (Tech. Rep. ISL-11). Palo Alto, CA: Xerox Parc.

VanLehn, K. (forthcoming-a) The representation and acquisition of procedural skills. Book in prep.

VanLehn, K. (forthcoming-b) The three inherent problems of induction.

VanLehn, K. & Brown, J.S. (1980) Planning Nets: A representation for formalizing analogies an semantic models of procedural skills. In R.E. Snow, P.A. Federico & W.E. Montague (eds.) *Aptitude, learning and instruction: Cognitive process analyses.* Hillsdale, NJ: Erlbaum.

VanLehn, K., Brown, J.S. & Greeno, J.G. (1984) Competitive argumentation in computational theories of cognition. In W. Kinsch, J. Miller & P. Polson (Eds.) *Methods and Tactics in Cognitive Science*, New York: Erlbaum.

Winston, P.H. Learning structural descriptions from examples. In P.H. Winston (ed.), *The Psychology of Computer Vision.* New York: McGraw-Hill, 1975.

Personnel Analysis Division,
AF/MPXA
5C360, The Pentagon
Washington, DC 20330

Air Force Human Resources Lab
AFHRL/MPD
Brooks AFB, TX 78235

AFOSR,
Life Sciences Directorate
Bolling Air Force Base
Washington, DC 20332

Dr. Robert Ahlers
Code N711
Human Factors Laboratory
NAVTRAEQUIPCEN
Orlando, FL 32813

Dr. Ed Aiken
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Earl A. Alluisi
HQ, AFHRL (AFSC)
Brooks AFB, TX 78235

Dr. John R. Anderson
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Steve Andriole
Perceptronics, Inc.
21111 Erwin Street
Woodland Hills, CA 91367-3713

Technical Director, ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Patricia Baggett
University of Colorado
Department of Psychology
Box 345
Boulder, CO 80309

Dr. Meryl S. Baker
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Gautam Biswas
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. John Black
Yale University
Box 11A, Yale Station
New Haven, CT 06520

Arthur S. Blaiwes
Code N711
Naval Training Equipment Center
Orlando, FL 32813

Dr. Jeff Bonar
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Richard Braby
NTEC Code 10
Orlando, FL 32751

Dr. Robert Breaux
Code N-095R
NAVTRAEQUIPCEN
Orlando, FL 32813

Dr. Ann Brown
Center for the Study of Reading
University of Illinois
51 Gerty Drive
Champaign, IL 61280

Dr. John S. Brown
XEROX Palo Alto Research
Center
3333 Coyote Road
Palo Alto, CA 94304

Dr. Bruce Buchanan
Computer Science Department
Stanford University
Stanford, CA 94305

Dr. Patricia A. Butler
NIE Mail Stop 1806
1200 19th St., NW
Washington, DC 20208

Dr. Robert Calfee
School of Education
Stanford University
Stanford, CA 94305

Dr. Jaime Carbonell
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Susan Carey
Harvard Graduate School of
Education
337 Gutman Library
Appian Way
Cambridge, MA 0138

Dr. Pat Carpenter
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Robert Carroll
NAVOP 01B7
Washington, DC 20370

Dr. Fred Chang
Navy Personnel R&D Center
Code 51
San Diego, CA 92152

Dr. Davida Charney
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Eugene Charniak
Brown University
Computer Science Department
Providence, RI 02912

Dr. Michelene Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Susan Chipman
Code 442PT
Office of Naval Research
800 N. Quincy St.
Arlington, VA 22217-5000

Mr. Raymond E. Christal
AFHRL/MOE
Brooks AFB, TX 78235

Dr. Yee-Yeen Chu
Perceptronics, Inc.
21111 Erwin Street
Woodland Hills, CA 91367-3713

Dr. William Clancey
Computer Science Department
Stanford University
Stanford, CA 94306

Scientific Advisor
to the DCNO (MPT)
Center for Naval Analysis
2000 North Beauregard Street
Alexandria, VA 22311

Chief of Naval Education
and Training
Liaison Office
Air Force Human Resource Laboratory
Operations Training Division
Williams AFB, AZ 85224

Assistant Chief of Staff
for Research, Development,
Test, and Evaluation
Naval Education and
Training Command (N-5)
NAS Pensacola, FL 32508

Dr. Allan M. Collins
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Stanley Collyer
Office of Naval Technology
800 N. Quincy Street
Arlington, VA 22217

CTB/McGraw-Hill Library
2500 Garden Road
Monterey, CA 93940

CDR Mike Curran
Office of Naval Research
800 N. Quincy St.
Code 270
Arlington, VA 22217-5000

Bryan Dallman
AFHRL/LRT
Lowry AFB, CO 80230

Dr. Charles E. Davis
Personnel and Training Research
Office of Naval Research
Code 442PT
800 North Quincy Street
Arlington, VA 22217-5000

Defense Technical
Information Center
Cameron Station, Bldg 5
Alexandria, VA 22314
Attn: TC
(12 Copies)

Dr. Thomas M. Duffy
Communications Design Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Edward E. Eddowes
CNATRA N301
Naval Air Station
Corpus Christi, TX 78419

Dr. John Ellis
Navy Personnel R&D Center
San Diego, CA 92252

Dr. Richard Elster
Deputy Assistant Secretary
of the Navy (Manpower)
Washington, DC 20350

Dr. Susan Embretson
University of Kansas
Psychology Department
Lawrence, KS 66045

Dr. Randy Engle
Department of Psychology
University of South Carolina
Columbia, SC 29208

Dr. William Epstein
University of Wisconsin
W. J. Brogden Psychology Bldg.
1202 W. Johnson Street
Madison, WI 53706

ERIC Facility-Acquisitions
4833 Rugby Avenue
Bethesda, MD 20014

Dr. K. Anders Ericsson
University of Colorado
Department of Psychology
Boulder, CO 80309

Edward Esty
Department of Education, CERI
MS 40
1200 19th St., NW
Washington, DC 20208

Dr. Beatrice J. Farr
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Marshall J. Farr
2520 North Vernon Street
Arlington, VA 22207

Dr. Pat Federico
Code 511
NPRDC
San Diego, CA 92152

Dr. Jerome A. Feldman
University of Rochester
Computer Science Department
Rochester, NY 14627

Dr. Paul Feltovich
Southern Illinois University
School of Medicine
Medical Education Department
P.O. Box 3926
Springfield, IL 62708

Mr. Wallace Feurzeig
Educational Technology
Bolt Beranek & Newman
10 Moulton St.
Cambridge, MA 02238

Dr. Craig I. Fields
ARPA
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Linda Flower
Carnegie-Mellon University
Department of English
Pittsburgh, PA 15213

Dr. Ken Forbus
Department of Computer Science
University of Illinois
Champaign, IL 61820

Dr. Carl H. Frederiksen
McGill University
3700 McTavish Street
Montreal, Quebec H3A 1Y2
CANADA

Dr. John R. Frederiksen
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

Dr. Norman Frederiksen
Educational Testing Service
Princeton, NJ 08541

Dr. R. Edward Geiselman
Department of Psychology
University of California
Los Angeles, CA 90024

Dr. Michael Genesereth
Stanford University
Computer Science Department
Stanford, CA 94305

Dr. Dedre Gentner
University of Illinois
Department of Psychology
603 E. Daniel St.
Champaign, IL 61820

Dr. Don Gentner
Center for Human
Information Processing
University of California
La Jolla, CA 92093

Dr. Robert Glaser
Learning Research
& Development Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Arthur M. Glenberg
University of Wisconsin
W. J. Brogden Psychology Bldg.
1202 W. Johnson Street
Madison, WI 53706

Dr. Marvin D. Glock
13 Stone Hall
Cornell University
Ithaca, NY 14853

Dr. Gene L. Gloye
Office of Naval Research
Detachment
1030 E. Green Street
Pasadena, CA 91106-2485

Dr. Sam Glucksberg
Princeton University
Department of Psychology
Green Hall
Princeton, NJ 08540

Dr. Joseph Goguen
Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dr. Sherrie Gott
AFHRL/MODJ
Brooks AFB, TX 78235

Dr. Richard H. Granger
Department of Computer Science
University of California, Irvine
Irvine, CA 92717

Dr. Wayne Gray
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. James G. Greeno
University of California
Berkeley, CA 94720

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Dr. David R. Lambert
Naval Ocean Systems Center
Code 441T
271 Catalina Boulevard
San Diego, CA 92152

Dr. Pat Langley
University of California
Department of Information
and Computer Science
Irvine, CA 92717

M. Diane Langston
Communications Design Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Kathleen LaPiana
Naval Health Sciences
Education and Training Command
Naval Medical Command,
National Capital Region
Bethesda, MD 20814-5022

Dr. Jill Larkin
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Robert Lawler
Information Sciences, FRL
GTE Laboratories, Inc.
40 Sylvan Road
Waltham, MA 02254

Dr. Paul E. Lehner
PAR Technology Corp.
7926 Jones Branch Drive
Suite 170
McLean, VA 22102

Dr. Alan M. Lesgold
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Jim Levin
University of California
Laboratory for Comparative
Human Cognition
D003A
La Jolla, CA 92093

Dr. Clayton Lewis
University of Colorado
Department of Computer Science
Campus Box 430
Boulder, CO 80309

Science and Technology Division
Library of Congress
Washington, DC 20540

Dr. Charlotte Linde
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dr. Marcia C. Linn
Lawrence Hall of Science
University of California
Berkeley, CA 94720

Dr. Don Lyon
P. O. Box 44
Higley, AZ 85236

Dr. Jane Malin
Mail Code SR 111
NASA Johnson Space Center
Houston, TX 77058

Dr. William L. Maloy (02)
Chief of Naval Education
and Training
Naval Air Station
Pensacola, FL 32508

Dr. Sandra P. Marshall
Department of Psychology
University of California
Santa Barbara, CA 93106

Dr. Manton M. Matthews
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. Richard E. Mayer
Department of Psychology
University of California
Santa Barbara, CA 93106

Dr. James McBride
Psychological Corporation
c/o Harcourt, Brace,
Javanovich Inc.
1250 West 6th Street
San Diego, CA 92101

Dr. James McMichael
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Barbara Means
Human Resources
Research Organization
1100 South Washington
Alexandria, VA 22314

Dr. Arthur Melmed
U. S. Department of Education
724 Brown
Washington, DC 20208

Dr. Al Meyrowitz
Office of Naval Research
Code 433
800 N. Quincy
Arlington, VA 22217-5000

Dr. George A. Miller
Department of Psychology
Green Hall
Princeton University
Princeton, NJ 08540

Dr. Lance A. Miller
IBM Thomas J. Watson
Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Dr. Andrew R. Molnar
Scientific and Engineering
Personnel and Education
National Science Foundation
Washington, DC 20550

Dr. William Montague
NPRDC Code 13
San Diego, CA 92152

Dr. Allen Munro
Behavioral Technology
Laboratories - USC
1845 S. Elena Ave., 4th Floor
Redondo Beach, CA 90277

Spec. Asst. for Research, Experimental & Academic Programs,
NTTC (Code 016)
NAS Memphis (75)
Millington, TN 38054

Dr. Richard E. Nisbett
University of Michigan
Institute for Social Research
Room 5261
Ann Arbor, MI 48109

Dr. Donald A. Norman
Institute for Cognitive Science
University of California
La Jolla, CA 92093

Director, Training Laboratory,
NPRDC (Code 05)
San Diego, CA 92152

Director, Manpower and Personnel
Laboratory,
NPRDC (Code 06)
San Diego, CA 92152

Director, Human Factors
& Organizational Systems Lab,
NPRDC (Code 07)
San Diego, CA 92152

Fleet Support Office,
NPRDC (Code 301)
San Diego, CA 92152

Library, NPRDC
Code P201L
San Diego, CA 92152

Commanding Officer,
Naval Research Laboratory
Code 2627
Washington, DC 20390

Dr. Ronald K. Hambleton
Laboratory of Psychometric and
Evaluative Research
University of Massachusetts
Amherst, MA 01003

Dr. Cheryl Hamel
NTEC
Orlando, FL 32813

Stevan Harnad
Editor, The Behavioral and
Brain Sciences
20 Nassau Street, Suite 240
Princeton, NJ 08540

Mr. William Hartung
PEAM Product Manager
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Wayne Harvey
SRI International
333 Ravenswood Ave.
Room B-S324
Menlo Park, CA 94025

Prof. John R. Hayes
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Barbara Hayes-Roth
Department of Computer Science
Stanford University
Stanford, CA 95305

Dr. Frederick Hayes-Roth
Teknowledge
525 University Ave.
Palo Alto, CA 94301

Dr. Joan I. Heller
Graduate Group in Science and
Mathematics Education
c/o School of Education
University of California
Berkeley, CA 94720

Dr. Geoffrey Hinton
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Jim Hollan
Code 51
Navy Personnel R & D Center
San Diego, CA 92152

Dr. John Holland
University of Michigan
2313 East Engineering
Ann Arbor, MI 48109

Dr. Melissa Holland
Army Research Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Keith Holyoak
University of Michigan
Human Performance Center
330 Packard Road
Ann Arbor, MI 48109

Dr. Ed Hutchins
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Dillon Inouye
WICAT Education Institute
Provo, UT 84057

Dr. S. Iyengar
Stanford University
Department of Psychology
Bldg. 4201 -- Jordan Hall
Stanford, CA 94305

Dr. Zachary Jacobson
Bureau of Management Consulting
365 Laurier Avenue West
Ottawa, Ontario K1A 0S5
CANADA

Dr. Robert Jannarone
Department of Psychology
University of South Carolina
Columbia, SC 29208

Dr. Claude Janvier
Directeur, CIRADE
Universite' du Quebec a Montreal
Montreal, Quebec H3C 3P8
CANADA

Margaret Jerome
c/o Dr. Peter Chandler
83, The Drive
Hove
Sussex
UNITED KINGDOM

Dr. Joseph E. Johnson
Assistant Dean for
Graduate Studies
College of Science and Mathematics
University of South Carolina
Columbia, SC 29208

Dr. Douglas H. Jones
Advanced Statistical
Technologies Corporation
10 Trafalgar Court
Lawrenceville, NJ 08148

Dr. Marcel Just
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Milton S. Katz
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Scott Kelso
Haskins Laboratories,
270 Crown Street
New Haven, CT 06510

Dr. Norman J. Kerr
Chief of Naval Education
and Training
Code 00A2
Naval Air Station
Pensacola, FL 32508

Dr. Dennis Kibler
University of California
Department of Information
and Computer Science
Irvine, CA 92717

Dr. David Kieras
University of Michigan
Technical Communication
College of Engineering
1223 E. Engineering Building
Ann Arbor, MI 48109

Dr. Peter Kincaid
Training Analysis
& Evaluation Group
Department of the Navy
Orlando, FL 32813

Dr. David Klahr
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Mazie Knerr
Program Manager
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. Janet L. Kolodner
Georgia Institute of Technology
School of Information
& Computer Science
Atlanta, GA 30332

Dr. Kenneth Kotovsky
Department of Psychology
Community College of
Allegheny County
800 Allegheny Avenue
Pittsburgh, PA 15233

Dr. Benjamin Kuipers
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139

Dr. Patrick Kyllonen
AFHRL/MOE
Brooks AFB, TX 78235

Dr. Harry F. O'Neil, Jr.
Training Research Lab
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Stellan Ohlsson
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Director, Technology Programs,
Office of Naval Research
Code 200
800 North Quincy Street
Arlington, VA 22217-5000

Director, Research Programs,
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Mathematics Group,
Office of Naval Research
Code 411MA
800 North Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 433
800 N. Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 442
800 N. Quincy St.
Arlington, VA 22217-5000

Office of Naval Research,
Code 442EP
800 N. Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 442PT
800 N. Quincy Street
Arlington, VA 22217-5000
(6 Copies)

Special Assistant for Marine
Corps Matters,
ONR Code 100M
800 N. Quincy St.
Arlington, VA 22217-5000

Psychologist
ONR Branch Office
1030 East Green Street
Pasadena, CA 91101

Dr. Judith Orasanu
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Jesse Orlansky
Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311

Prof. Seymour Papert
20C-109
Massachusetts Institute
of Technology
Cambridge, MA 02139

Lt. Col. (Dr.) David Payne
AFHRL
Brooks AFB, TX 78235

Dr. Douglas Pearse
DCIEM
Box 2000
Downsview, Ontario
CANADA

Dr. Nancy Pennington
University of Chicago
Graduate School of Business
1101 E. 58th St.
Chicago, IL 60637

Military Assistant for Training and
Personnel Technology,
OUSD (R & E)
Room 3D129, The Pentagon
Washington, DC 20301

Dr. David N. Perkins
Educational Technology Center
337 Gutman Library
Appian Way
Cambridge, MA 02138

Administrative Sciences Department,
Naval Postgraduate School
Monterey, CA 93940

Department of Operations Research,
Naval Postgraduate School
Monterey, CA 93940

Department of Computer Science,
Naval Postgraduate School
Monterey, CA 93940

Dr. Tjeerd Plomp
Twente University of Technology
Department of Education
P.O. Box 217
7500 AE ENSCHEDE
THE NETHERLANDS

Dr. Martha Polson
Department of Psychology
Campus Box 346
University of Colorado
Boulder, CO 80309

Dr. Peter Polson
University of Colorado
Department of Psychology
Boulder, CO 80309

Dr. Steven E. Poltrock
MCC
9430 Research Blvd.
Echelon Bldg #1
Austin, TX 78759-6509

Dr. Harry E. Pople
University of Pittsburgh
Decision Systems Laboratory
1360 Scaife Hall
Pittsburgh, PA 15261

Dr. Joseph Psotka
ATTN: PERI-IC
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Lynne Reder
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. James A. Reggia
University of Maryland
School of Medicine
Department of Neurology
22 South Greene Street
Baltimore, MD 21201

Dr. Fred Reif
Physics Department
University of California
Berkeley, CA 94720

Dr. Lauren Resnick
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Mary S. Riley
Program in Cognitive Science
Center for Human Information
Processing
University of California
La Jolla, CA 92093

Dr. Andrew M. Rose
American Institutes
for Research
1055 Thomas Jefferson St., NW
Washington, DC 20007

Dr. William B. Rouse
Georgia Institute of Technology
School of Industrial & Systems
Engineering
Atlanta, GA 30332

Dr. Donald Rubin
Statistics Department
Science Center, Room 608
1 Oxford Street
Harvard University
Cambridge, MA 02138

Dr. Lawrence Rudner
403 Elm Avenue
Takoma Park, MD 20012

Dr. Michael J. Samet
Perceptronics, Inc
6271 Variel Avenue
Woodland Hills, CA 91364

Dr. Robert Sasmor
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Roger Schank
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Dr. Alan H. Schoenfeld
University of California
Department of Education
Berkeley, CA 94720

Dr. Janet Schofield
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Judith Segal
Room 819F
NIE
1200 19th Street N.W.
Washington, DC 20208

Dr. Ramsay W. Selden
NIE
Mail Stop 1241
1200 19th St., NW
Washington, DC 20208

Dr. Michael G. Shafto
ONR Code 442PT
800 N. Quincy Street
Arlington, VA 22217-5000

Dr. Sylvia A. S. Shafto
National Institute of Education
1200 19th Street
Mail Stop 1806
Washington, DC 20208

Dr. T. B. Sheridan
Dept. of Mechanical Engineering
MIT
Cambridge, MA 02139

Dr. Ted Shortliffe
Computer Science Department
Stanford University
Stanford, CA 94305

Dr. Lee Shulman
Stanford University
1040 Cathcart Way
Stanford, CA 94305

Dr. Miriam Shustack
Code 51
Navy Personnel R & D Center
San Diego, CA 92152

Dr. Robert S. Siegler
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Herbert A. Simon
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Zita M Simutis
Instructional Technology
Systems Area
ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. H. Wallace Sinaiko
Manpower Research
and Advisory Services
Smithsonian Institution
801 North Pitt Street
Alexandria, VA 22314

Dr. Derek Sleeman
Stanford University
School of Education
Stanford, CA 94305

Dr. Edward E. Smith
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Alfred F. Smode
Senior Scientist
Code 7B
Naval Training Equipment Center
Orlando, FL 32813

Dr. Richard Snow
Liaison Scientist
Office of Naval Research
Branch Office, London
Box 39
FPO New York, NY 09510

Dr. Elliot Soloway
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Dr. Richard Sorensen
Navy Personnel R&D Center
San Diego, CA 92152

James J. Staszewski
Research Associate
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Marian Stearns
SRI International
333 Ravenswood Ave.
Room B-S324
Menlo Park, CA 94025

Dr. Robert Sternberg
Department of Psychology
Yale University
Box 11A, Yale Station
New Haven, CT 06520

Dr. Albert Stevens
Bolt Beranek & Newman, Inc.
10 Moulton St.
Cambridge, MA 02238

Dr. Paul J. Sticha
Senior Staff Scientist
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. Thomas Sticht
Navy Personnel R&D Center
San Diego, CA 92152

Dr. David Stone
KAJ Software, Inc.
3420 East Shea Blvd.
Suite 161
Phoenix, AZ 85028

Cdr Michael Suman, PD 303
Naval Training Equipment Center
Code N51, Comptroller
Orlando, FL 32813

Dr. Hariharan Swaminathan
Laboratory of Psychometric and
Evaluation Research
School of Education
University of Massachusetts
Amherst, MA 01003

Mr. Brad Sympson
Navy Personnel R&D Center
San Diego, CA 92152

Dr. John Tangney
AFOSR/NL
Bolling AFB, DC 20332

Dr. Kikumi Tatsuoka
CERL
252 Engineering Research
Laboratory
Urbana, IL 61801

Dr. Maurice Tatsuoka
220 Education Bldg
1310 S. Sixth St.
Champaign, IL 61820

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1185 Coleman Avenue, Box 580
Santa Clara, CA 95052

Dr. Douglas Towne
Behavioral Technology Labs
1845 S. Elena Ave.
Redondo Beach, CA 90277

Dr. Amos Tversky
Stanford University
Dept. of Psychology
Stanford, CA 94305

Dr. James Tweeddale
Technical Director
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Paul Twohig
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. J. Uhlaner
Uhlaner Consultants
4258 Bonavita Drive
Encino, CA 91436

Headquarters, U. S. Marine Corps
Code MPI-20
Washington, DC 20380

Dr. Kurt Van Lehn
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304

Dr. Beth Warren
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Edward Wegman
Office of Naval Research
Code 411
800 North Quincy Street
Arlington, VA 22217-5000

Dr. David J. Weiss
N660 Elliott Hall
University of Minnesota
75 E. River Road
Minneapolis, MN 55455

Dr. Keith T. Wescourt
FMC Corporation
Central Engineering Labs
1185 Coleman Ave., Box 580
Santa Clara, CA 95052

Dr. Douglas Wetzel
Code 12
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Barbara White
Bolt Beranek & Newman, Inc.
10 Moulton Street
Cambridge, MA 02238

Dr. Hilda Wing
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Robert A. Wisher
U.S. Army Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Martin F. Wiskoff
Navy Personnel R & D Center
San Diego, CA 92152

Dr. Frank Withrow
U. S. Office of Education
400 Maryland Ave. SW
Washington, DC 20202

Dr. Merlin C. Wittrock
Graduate School of Education
UCLA
Los Angeles, CA 90024

Mr. John H. Wolfe
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Wallace Wulfeck, III
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Joe Yasatuke
AFHRL/LRT
Lowry AFB, CO 80230

Mr. Carl York
System Development Foundation
181 Lytton Avenue
Suite 210
Palo Alto, CA 94301

Dr. Joseph L. Young
Memory & Cognitive
Processes
National Science Foundation
Washington, DC 20550

Dr. Steven Zornetzer
Office of Naval Research
Code 440
800 N. Quincy St.
Arlington, VA 22217-5000

Dr. Michael J. Zyda
Naval Postgraduate School
Code 52CK
Monterey, CA 93943

DTIC

FILMED

4-86

END