Human Aspects of Computing

H. Ledgard Editor

Psychology of Calculator Languages: A Framework for Describing Differences in Users' Knowledge

Richard E. Mayer and Piraye Bayman
University of California, Santa Barbara

This paper presents a framework for describing users' knowledge of how a simple four-function calculator operates. Differences among novices and experts in their conceptions of "what goes on inside the calculator" for various sequences of button presses are summarized. Individual differences include different views on when an expression is evaluated, different procedures for evaluating a chain of arithmetic, and different rules for evaluating unusual sequences of key presses.

Key Words and Phrases: calculator, learning, instruction, psychology

CR Categories: 1.50, 2.12, 4.20

I. Introduction

A major goal of this article is to provide a framework for describing users' knowledge of calculator language, i.e., users' intuitions concerning the underlying logic of a simple four-function calculator when a series of buttons are pressed. Another goal is to use this technique for pinpointing some of the differences among actual users in their knowledge of calculator language. The first section of this paper provides a rationale and a brief literature review. The second section describes the transaction approach for analyzing calculator language. The third section summarizes a study of individual differences among users in their knowledge of calculator language. The final section provides a summary and a set of tentative recommendations.

This work was supported by Grant NIE-G-80-0118 from the National Institute of Education, Program in Teaching and Learning. Requests for reprints should be sent to Richard E. Mayer, Department of Psychology, University of California, Santa Barbara, CA 93106.

Authors' present address: R.E. Mayer and P. Bayman, Department of Psychology, University of California, Santa Barbara, CA 93106.

II. Rationale

A. Performance vs Competence

The traditional distinction between performance and competence can and should be applied to users' learning of calculator language. Performance, of course, refers to what the user can *do*, such as compute answers for a class of problems; competence refers to what the user *knows*, such as the user's mental model of the calculator.

It is possible for two users to give identical answers to simple arithmetic computation problems but possess vastly different underlying knowledge of calculator language. We found that a subject would claim that to find the answer for 22 × 114, the calculator simply "looks up" the answer for that problem in its memory. Another subject assumed that the calculator used "internal registers," and followed certain "control procedures." For 22 × 114, the calculator would store the numbers 22 and 114 in memory and would use the multiplication algorithm to work on them. These two subjects seem to have had different "mental models" for the calculator. Thus, for a complete description of "what is learned" by different users, we must be able to describe the users' competence as well as their performance. Similarly, Greeno [3] has argued for emphasis on cognitive objectives of instruction rather than focusing solely on behavioral objectives of instruction.

B. Black Box vs Glass Box

A second important distinction concerns learning by memorizing vs learning by understanding [15]. When we apply this distinction to the learning of calculator language, we can point to the difference between the "black box approach" and the "glass box approach." In the black box approach to learning calculator language, the user focuses only on the external features of calculator language—you put in a sequence of key presses and out comes the answer as if by magic. The operations inside the calculator are hidden from the user, forcing the user to treat the calculator as a black box that cannot be understood. A user who learns by the black box method is forced to memorize sequences of key presses for each type of problem, without understanding what the key presses actually mean. For example, some manuals describe how to use a constant. Let us say you want to multiply a set of numbers by 2.3. The manual may instruct you to enter the sequence 2.3 × ×; then, for any number you want multiplied by 2.3 you just enter that number followed by an equals (=). Although memorized procedures, such as the constant sequence, may work in the sense that they generate the desired answer, the user is not able to relate the sequence of key strokes to an understanding of what goes on inside the calculator. DuBoulay and O'Shea [1] have noted a similar phenomenon with respect to children learning LOGO; some users act as if the internal operations of the machine are hidden and not understandable.

In the glass box approach [2], the user is able to see how a sequence of key strokes is related to changes in the internal state of the calculator and how these changes are related to the final answer. Each command—in this case, each key stroke—results in some change inside the calculator, and these changes can be described and understood. The user who learns by the glass box approach may be able to describe why the constant procedure works by describing the nature of internal displays and incrementing operations.

The level of description of events in the glass box approach need not, indeed, should not, be at the "blood and guts" level. By this we mean that users need not become electronics experts. There is an appropriate level of description that Young [16] refers to as the user's mental model of the calculator: "For an interactive system to be satisfactory, it is important that its intended users be able to form a model of the system which enables them to predict its behavior."

DuBoulay, O'Shea, and Monk [2] have suggested that novices be exposed to a notational machine, i.e., ". . .an idealized model of the computer implied by the constraints of the programming languages. . ." and which is analogous to ". . .other mechanisms with which the novice is more familiar. . . ." As an example, DuBoulay and O'Shea [1] have developed a "LOGO machine" to represent the internal actions that occur for LOGO statements. Further, DuBoulay, O'Shea, and Monk [2] have offered two important properties for selecting a model that clarifies the hidden operations of a language: (1) simplicity—there should be a "small number of parts that interact in ways that can be easily understood," and (2) visibility—novices should be able to see "selected parts and processes of this notational machine in action."

As an example, let us suppose that we want students to learn how to solve simple arithmetic problems with a calculator. We could give them plenty of hands-on experience, without any guidance as to what goes on inside the calculator until they were all able to solve simple problems. However, Scandura, Lowerre, Veneski, and Scandura [13] found that students who taught themselves to use calculators often developed bizarre intuitions; one student, for example, concluded that the plus (+) and equals (=) keys did nothing since they caused no visible change in the display. Instruction that emphasizes the understanding of how the machine operates on a sequence of button presses might provide a better base on which to build *further computer* concepts.

What are the benefits of instruction that foster glass box learning rather than black box learning? Past research by Gestalt psychologists [15] suggests that learning by understanding, such as in the glass box procedure, leads to superior long-term retention and superior transfer to novel problems. In addition, the glass box approach may influence attitudes concerning the understandability of computers and calculators. Although there is promising support for these assertions in studies of how novices learn simple programming languages [4–7,9], much more research is needed concerning the role of glass box instruction for calculators.

C. Computer literacy

The previous sections have presented two ways of describing what is learned (i.e., performance vs competence) and two ways of teaching the use of calculators (i.e., black box vs glass box). Why is it important to focus on how students learn and represent knowledge about calculators? The reason is that calculators (as well as electronic games) usually involve a user's first exposure to a computational machine and a language. Thus, calculators provide the first step in the development of a user's computer literacy—the understanding of how to interact with computational machines.

In addition, calculators have become a part of society, infiltrating the home, work, and school lives of ordinary people [10]. Teachers [11] have recognized calculators as necessary tools in our society: "The National Council of Teachers of Mathematics recommends that mathematics programs take full advantage of the power of calculators and computers at all grade levels." However, in spite of the potential for using calculators as the first step towards computer literacy, there is also the potential that they will be used as tools whose operations must be blindly memorized. For example, DuBoulay, O'Shea, and Monk [2] recently pointed out: "The manuals accompanying certain makes of pocket calculators make no attempt to explain the reason why given sequences of button presses carry out the given computations. The user must follow the manual's instructions blindly because it is difficult for him to imagine what kind of underlying machine could be inside that demands these particular sequences of presses. During the course of a calculation, he has to guess the current state of the device . . . because the device gives little or no external indications of its internal state."

Unfortunately, the research community has been very slow in providing information that would be useful in this impending calculator-curriculum revolution. For example, most experimental studies have been concerned with whether using calculators in the classroom affects overall achievement and/or attitude in mathematics (see [12, 14]); but as Roberts [12] recently concluded, ". . .the research literature offers no guidance. . ." concerning how to incorporate calculators into school curricula.

The development of a theory of how users conceptualize calculator language has implications for the design of calculator languages, for instructional procedures, and for integration of calculators into school curricula. This paper is based on the idea that calculators are here to stay, that large numbers of ordinary (nonprogrammers) people will be using them, and that calculators provide most users with their first introduction to computer concepts. Such users will inevitably develop attitudes and approaches to human/computer interaction in

Table I. Sixteen Elementary Calculator Commands.

Name	Command	Example	Description
P1	# after #	2 3	Pressing a number key after pressing a number key
P2	# after +	+ 3	Pressing a number key after pressing a plus key
P3	# after =	= 3	Pressing a number key after pressing an equals key
P4	+ after #	2 +	Pressing a plus key after pressing a number key
P5	+ after +	+ +	Pressing a plus key after pressing a plus key
P6	+ after =	= +	Pressing a plus key after pressing an equals key
P7	= after #	3 =	Pressing an equals key after pressing a number key
P8	= after +	+ =	Pressing an equals key after pressing a plus key
P9	= after =	= =	Pressing an equals key after pressing an equals key
P10	# after ×	× 3	Pressing a number key after pressing a times key
P11	× after #	2 ×	Pressing a times key after pressing a number key
P12	= after ×	× =	Pressing an equals key after pressing a times key
P13	× after =	= ×	Pressing a times key after pressing an equals key
P14	× after ×	× ×	Pressing a times key after pressing a times key
P15	+ after ×	× +	Pressing a plus key after pressing a times key
P16	× after +	+ ×	Pressing a times key after pressing a plus key

the course of learning to use their calculator even if the users are self-taught. This paper provides some information that may be relevant to understanding what intuitions individual users have about calculators.

III. A Transaction Analysis of Calculator Language

The goal of this section is to develop an appropriate level of describing what happens inside the calculator for each type of key press, based on DuBoulay, O'Shea, and Monk's [2] criteria of simplicity and visibility. In particular, the transaction approach is applied to the operating system[1] of electronic calculators, or to what can be called calculator language. The goal is not to provide a formal, mathematical representation of the calculator's operating system, but to provide an idealized model of the calculator that can be used to describe users' knowledge and to help novices understand calculator language. It should be pointed out that the transaction approach may serve both as (1) a descriptive model of the users' knowledge of calculator language, and (2) a prescriptive model for curriculum development. This paper presents data concerning the first implementation but also suggests implications concerning the second.

For the purposes of this analysis we assume that each user's conception of calculator language can be specified as a set of productions, or condition–action pairs. The condition refers to some key press (i.e., some command) and the action refers to one or more transactions (i.e., an operation applied to an object at a location in the calculator). Thus, the transaction approach involves locating the transaction (or list of transactions) that a user associates with a given command.

A previous paper [6] summarized a conceptual analysis of BASIC that emphasized "transactions" for describing the language in a simple and visible way. A model was constructed that consisted of a ticket window to represent the input function, a note pad to represent the output function, a memory scoreboard to represent the memory function, a scratch pad to represent the logic and arithmetic function, a shopping list with pointer arrow to represent executive control. Each elementary BASIC statement was described as a list of *transactions*, and each transaction consisted of some *operation* applied to some *object* at some *location* in the computer. Using a small collection of transactions it was possible to describe each of the elementary BASIC statements. Further, there is substantial evidence that instruction in BASIC which emphasizes the transaction level of description results in superior performance in creative program writing and interpreting written programs [4, 5, 7, 9].

The relevant conditions (or commands) for the present analysis are based on pressing keys on the calculator's keyboard. The keys relevant to a very simple four-function calculator are number keys (i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9), operation keys (i.e., +, −, ×, ÷), equals key (=), decimal key (.), and clear key (CLR). We assumed that the calculator used arithmetic logic (rather than algebraic or reverse Polish notation), and we focused on only three number keys (i.e., 2, 3, and 7), two operation keys (i.e., + and ×), and the equals key (=).

Thus, at first blush it seems the basic conditions are each of the single key presses, such as pressing a number key, pressing an operation key, and so on. However, interviews with users suggest that key presses have different meanings depending on the immediately preceding key press; for example, pressing a plus key after pressing a number key has a different effect than pressing a plus key after pressing an equals key, for some users. Thus, the conditions (or user commands) can be listed as some key being pressed given that some key was pressed immediately before. For example, typical commands in the present analysis are listed in Table I. There are certainly many other possible commands, but we have focused on this set of 16 elementary calculator commands as an example.

[1] We use the term "operating system" to refer to a program that establishes the mode of user–machine interaction and provides for efficient control of system components. Thus, in the present article, the terms "control program" or "instruction set" could be substituted for operating system. The term in this article does not fit the strict definition of "operating system", i.e., a system for mediating among the demands of multiple users in a time sharing system.

To describe the actions that occur for any command, the transaction approach [6] requires that we specify the triplet of location, object, and operation. The possible locations within the calculator are as follows:

(1) *Display.* The external display normally consists of at least eight spaces, where a place can hold one digit. The display fills from the right.
(2) *Register.* An internal register is inside the calculator and consists of a series of subregisters that hold individual numbers and operators. Expressions are held in the order of input, with the first number of the left, followed by first operator, and with new numbers and operators entered to the right of existed filled subregisters.
(3) *Keyboard.* The external set of keys includes number, operation, and equals keys.

The possible objects include:

(1) *Number.* A number is any single or multiple digit sequence such as 2, 14, or 156.
(2) *Operation.* An operation is a mathematical symbol for some arithmetic computation such as addition (+) or multiplication (×).
(3) *Expression.* An expression is a sequence consisting of numbers and operators such as $2 + 3$ or $2 +$ or $2 + 3 \times 7$.

Some operations that are relevant to computer language are as follows:

(1) *Find.* Locate a particular object; e.g., find a number that was just entered from the keyboard.
(2) *Destroy.* A number or expression is erased from the display or register; e.g., when you press the equals key the previous number in the display is erased (and replaced with a new one).
(3) *Create.* A number or expression is placed in a display or register; e.g., when you press a number key that number appears in the display.
(4) *Evaluation.* An expression from the register is converted into a single number using the rules of arithmetic; e.g., the evaluation of $3 + 2$ is 5. (For the current discussion, evaluation of a number or numbers followed by an operation is the number; i.e., evaluation of 3 is 3 or evaluation of $3 +$ is 3). It is also possible to evaluate expressions from the register and display together; for example, the evaluation of $2 +$ in the register and 3 in the display may be 5.

Table II gives a summary of some typical actions that might occur with the calculator. Each is expressed as a list of transactions; for example, D = R consists of four separate transactions, while D = D requires only one. A user's conception of what a particular command means can be expressed as a production; for example, the production,

P2 If # after + Then D = # and R = "R + #"

means that when the number key is pressed after a plus key, the user assumes that the calculator executes the four transactions for D = # and the three transactions for R = "R + #" as listed in Table II. Thus, for the sequence 7 + 3, when the 3 key is pressed, the display is changed to 3 and the register's expression is changed to "7 + 3". A user's intuitions concerning calculation language can thus be expressed as a list of productions such as the one given above.

IV. Empirical Studies

There has not been adequate research concerning how students come to understand the operation of calculators. As noted earlier, almost all behavioral research concerning calculators has been directed at the gross issue of whether the availability of calculators in the classroom has any effect on mathematics achievement or attitude (see[11, 14]). The present study addresses a different issue, namely, what types of hypotheses do people have concerning how calculators operate? The goal of this research is to determine whether the transaction approach can be successfully used as a framework for describing differences in users' knowledge. The goal of these studies is not to test the transaction approach as a "theory," since it is used here only as a framework for describing what is learned. For more detail concerning the methodology and data analyses, see Mayer and Bayman [8].

A. Method

Our study involved 33 college students who had no computer programming experience ("novices") and 33 college students who were enrolled in advanced programming courses ("experts").[2] Subjects participated in order to fullfill a course requirement. Each student was given a four page questionnaire with 88 problems. Each problem listed a series of key presses and asked the student to predict what number would be in the display, assuming a standard four-function calculator.was being used. In addition, subjects were given a questionnaire asking how many hours a week they used a calculator, how many years they had been using a calculator, what kind of calculators they knew, and which calculator model(s) they owned, if any.

B. Standard Sequences: When to Evaluate

Our subjects differed greatly with respect to when

[2] The main difference between experts and novices is that all the experts had formal instruction in computer programming and had some introduction to operating systems, while none of the novices did. As might be expected, there were other demographic differences between the groups: experts were older, $t(60) = 2.66, p < 0.01$, and experts scored higher in SAT-Mathematics, $t(44) = 4.67, p < 0.001$. Thus, while the main comparison was between "liberal arts" students who had no formal programming experience and "engineering" students who had formal training in programming, any comparisons between the two groups must be made in light of other group differences such as age and SAT scores. Individual analysis of the performance of novices who scored high on the SAT revealed that they did not perform any more like the experts than did the novices who scored low on the SAT.

Table II. Some Possible Transactions in Computer Language.

Transaction	Location	Object	Operation	Description
D = D	display	number	no change	No change in the display
D = 3	display	number	find	Find the old number in the display
	display	number	destroy	Erase it
	keyboard	number	find	Find the number that has been entered in the keyboard
	display	number	create	Put new number in display
D = R	display	number	find	Find the old number in the display
	display	number	destroy	Erase it
	register	number	find	Find the number currently in the register (but do not destroy it)
	display	number	create	Copy the number from the register into the display
D = eval (R)	display	number	find	Find the old number in the display
	display	number	destroy	Erase it
	register	expression	find	Find the expression in the register
	register	expression	evaluate	Evaluate the expression in the register (but do not destroy it)
	display	number	create	Put the evaluated value of the register in the display
D = eval (D + R)	display	number	find	Find the number currently in the display
	register	number	find	Find the number currently in the register (but do not alter it)
	register	expression	evaluate	Evaluate the value of the display plus the register
	display	number	create	Put the new sum in the display
R = R	register	expression	no change	No change in the register
	register	expression	find	Find the old expression in the register
R = #	register	expression	destroy	Erase the old expression from the register
	keyboard	number	find	Find the number that has been entered in the keyboard
	register	number	create	Put the new number from the keyboard in the register
R = "R +"	register	expression	find	Retain the existing expression that is in the register
	register	operator	create	Place a plus sign to the right of the expression in the register
R = "R + #"	register	expression	find	Retain the existing expression that is in the register
	keyboard	number	find	Find the number that has just been entered in the keyboard
	register	number	create	Place the number to the right of the expression in the register
R = eval (R)	register	expression	find	Find the current expression or number in the register
	register	expression	evaluate	Evaluate the expression or number
	register	expression	destroy	Erase the expression from the register
	register	number	create	Replace it with the evaluation of the old number or expression
R = eval (D + R)	display	number	find	Find the number currently in the display
	register	number	find	Find the number currently in the register
	register	expression	evaluate	Add them together
	register	number	create	Replace it with the sum
R = eval (R + R)	register	number	find	Find the number in the register
	register	expression	evaluate	Add the number to itself
	register	number	destroy	Erase the old number from the register
	register	number	create	Replace it with the new sum
R = O	register	expression	find	Find the existing number or expression in the register
	register	expression	destroy	Erase that expression or number
	register	number	create	Replace it with zero

they thought an expression should be evaluated. For example, consider the sequence 2 + 3. Some subjects answered "3" and some answered "5". Those who gave 5 seem to be using what we called "immediate evaluation" for # after +. Whenever a number key is pressed after a plus key, the entire expression is evaluated and displayed. However, those who gave 3 as an answer seem to be using "delayed evaluation" for # after +. They wait for some other key press (such as an equals or a plus or a multiply) before they evaluate and display. How would a subject predict the calculator would respond to 2 + 3 + 7 ? The answer was 12 for the immediate evaluators and 7 for the subjects who relied on delayed evaluation for # after +. Our subjects were very consistent, although experts were significantly more consistent than novices in such judgments.

Now consider the problem 2 + 3 +. Some subjects gave 5 as the answer while others gave 3. Those who gave 3 act as if there is delayed evaluation for + after 3. For those who gave 5 as an answer, if they also gave 5 as an answer to problems like 2 + 3, they are not evaluating for + after # either; however, if they gave 3 as an answer for 2 + 3, then they seem to opt for "immediate evaluation" for + after #. Similarly, a se-

Table III. Three Major Conceptions of When to Evaluate An Expression.

	Example		Proportion of subjects	
Conception	Problem	Answer	Novices	Experts
Evaluate as soon as a number key is pressed	2 + 3 2 + 3 + 2 + 3 =	5 5 5	0.21	0.06
Evaluate as soon as an operation key is pressed	2 + 3 2 + 3 + 2 + 3 =	3 5 5	0.39	0.73
Evaluate as soon as an equals key is pressed	2 + 3 2 + 3 + 2 + 3 =	3 3 5	0.39	0.21

For 2 × 2 contingency table, $\chi^2 = 7.44$, df = 1, $p < 0.01$

quence like 2 × 3 + results in 6 for subjects who rely on immediate evaluation for + after # but in 3 for those who rely on delayed evaluation for + after #. (Note that if our subject relies on immediate evaluation for # after ×, then the answer will also be 6.)

Finally, consider the problem 2 + 3 =. All subjects gave 5 as an answer. Or consider the problem 2 + 3 + 7 =. All subjects gave 12 as an answer. However, if our subjects were using delayed evaluation for # after + and delayed evaluation for + after #, then we know they were waiting for an equals sign before they evaluate; thus, these subjects would opt for immediate evaluation for = after #.

Based on a systematic analysis of our subjects' performance, we noted three basic strategies for determining when to evaluate an expression: for a number key, for an operation key, or for an equals key. These concepts are summarized in Table III. The consensus of the experts is that a calculator should evaluate when an operation key is pressed after a number, as is common in most but not all calculators. Novices tend to have much more diverse conceptions which are significantly different from experts.[3] It may also be pointed out that we found no relation between the conceptions of our subjects and the operating systems of their own calculators, nor between the conceptions of our subjects and the amount of time spent each week with a calculator.[4]

C. Standard Sequences: Chains of Arithmetic

How would you predict a calculator would answer 2 + 3 × 7 = ? How about the problem 2 × 3 + 7 = ? Our subjects varied with respect to how they evaluated a chain of arithmetic. The vast majority of subjects executed the operations in order from left to right, yielding answers of 35 and 13, respectively, for the above problems. Some subjects tended to opt for multiplication being carried out before addition, yielding answers of 23 and 13, respectively. Some subjects tended to opt for addition being carried out before multiplication, yielding 35 and 20, respectively. Some subjects opted for carrying out the second operation first, yielding 23 and 20, respectively. Finally, some subjects simply ignored all but the last computation, yielding 21 and 10, respectively. Table IV summarizes the major strategies for evaluating a chain, based on an analysis of each subject's performance on several problems. As can be seen, most subjects opted for left-to-right evaluation of a chain, although a substantial minority of experts assumed multiplications were carried out before addition. This procedure is characteristic of some sophisticated calculators and computer commands.

Table IV. Five Conceptions of How to Evaluate an Arithmetic Chain.

	Example		Proportion of subjects	
Conception	Problem	Answer	Novices	Experts
Evaluate in order from left to right	2 + 3 × 7 = 2 × 3 + 7 =	35 13	0.88	0.70
Evaluate backwards from right to left	2 + 3 × 7 = 2 × 3 + 7 =	23 20	0.03	0.00
Evaluate only the last computation	2 + 3 × 7 = 2 × 3 + 7 =	21 10	0.03	0.00
Evaluate multiplication before addition	2 + 3 × 7 = 2 × 3 + 7 =	23 13	0.03	0.30
Evaluate addition before multiplication	2 + 3 × 7 = 2 × 3 + 7 =	35 20	0.03	0.00

For 2 × 2 contingency table, $\chi^2 = 6.98$, df = 1, $p < 0.01$

D. Nonstandard Sequences: Equals after Operator

The foregoing two sections demonstrated that there are considerable differences among subjects' interpretations of calculator operations even for standard sequences of key strokes. A standard sequence is defined as one that begins with a number and in which an operator (like + or ×) or equals (=) may only follow a number. In the present section we explore subjects' conceptions of how the calculator responds to nonstandard

[3] The categorization of subjects, as indicated in Tables III through VII, was based on an analysis of all the problems (i.e, 88 responses) rather than just the few examples given in the text of this report. Subjects were classified using a forced choice procedure, so that each subject was placed in the category that was most consistent with the data that he or she provided us with. Chi square tests were conducted on the data in each of the Tables III through VII, using a 2 × 2 contingency table and Yates corrective. The expected frequencies were based on the overall mean for each category and tested the null hypothesis that there was no difference between experts and novices in the pattern of category frequencies.

[4] For example, the most frequently owned calculators were Texas Instruments, Rockwell, and Sharp. The answers given by each of these calculators for each of the 88 problems were compared to the answers given by each subject. Difference scores were computed by counting the number of times that the subject gave an answer that was different from a given brand. For novices who owned TI calculators the difference score was the lowest for TI (8.0) and the highest for the Rockwell (20.0), Sharp (14.1) models being in between. However, for students who owned calculators other than TIs the same pattern was obtained, with the lowest difference score for TI (9.8) and higher scores for Rockwell (2.16) and Sharp (15.8). A similar pattern was obtained among experts: for TI owners and nonowners, their predictions most closely corresponded to the performance of a TI calculator rather than other brands. Analyses of variance indicated no differences between TI owners and owners of other brands.

Table V. Three Major Conceptions of How to Evaluate Equals After Operator.

	Example		Proportion of subjects	
Conception	Problem	Answer	Novices	Experts
Ignore the non-standard sequence	7 + = 7 × =	7 7	0.82	0.76
Reset the display	7 + = 7 × =	0 0	0.09	0.06
Increment the display	7 + = 7 × =	14 49	0.09	0.18

For 2 × 2 contingency table, $\chi^2 = 0.52$, df = 1, p = *n.s.*

Table VI. Three Major Conceptions of How to Evaluate Two Consecutive Operators.

	Example		Proportion of subjects	
Conception	Problem	Answer	Novices	Experts
Ignore the non-standard sequence	2 ++ = 2 ×× =	2 2	0.85	0.73
Reset the display	2 ++ = 2 ×× =	0 0	0.03	0.00
Increment the display	2 ++ = 2 ×× =	6 or 8 8 or 16	0.12	0.27

For 2 × 2 contingency table, $\chi^2 = 1.53$, df = 1, p = *n.s.*

sequences of key strokes. A nonstandard sequence violates the above "grammatical rule of arithmetic" by having two or more operators (+ or ×) and/or equal sign (=) in sequence. Users' predictions concerning nonstandard sequences are useful because they allow us to diagnose users' conceptions of the internal operation of the calculator.

For example, consider the sequences 7 + = or 7 × =. How do subjects interpret the calculators' operations? Some subjects assume that a nonstandard sequence results in the display being reset; for example, if resetting the display means setting it to zero, then subjects give 0 as the answer to the above problems. Another version of the reset strategy is to assume that the calculator will show an E in the display, or that it will flash on and off. A second group of subjects act as if the calculator simply ignores the nonstandard sequence; in this case, the calculator display has 7 in it for each of the above sequences. Finally, a third major group acts as if a number has been inserted between the operator and the display; for example, they treat 7 + = as 7 + 7 = and give an answer of 14, or they treat 7 × = as 7 × 7 = and give an answer of 49. We call these subjects "incrementing display" subjects because they act as if the number in the display is added to the number in the internal register. There are several variations on the incrementing display strategy; for example, 2 + 3 + = can result in 10 or in 8 depending on the subject's conception of when evaluation occurs.

Table V summarizes these three major concepts of what happens when equals follows an operation; as can be seen, the strategy of ignoring the nonstandard sequence is the most common, but experts are far more likely to opt for the incrementing display conceptualization. The incrementing procedure is a feature of some more sophisticated calculators and reflects a more sophisticated understanding of internal registers.

E. Nonstandard Sequences: Two Consecutive Operators

Another nonstandard sequence is to have two consecutive operators, such as 2 + + = and 2 × × =. The same strategies were obtained as in the previous section. One subject thought the display would be reset, for example, the answers would be 0 for each problem. Some subjects ignored the nonstandard sequence; thus, the display would say 2 for each problem. For example, 2 + + was treated as if it were 2 + and hence 2 would be displayed. Finally, some subjects used an incrementing strategy; for example, 2 ++ = could be interpreted as 2 + 2 + 2 =, thus yielding an answer of 6, and 2 × × = could be interpreted as 2 × 2 × 2 =, yielding an answer of 8. A variation of this strategy is to treat 2 + + = as 2 + 2 = 4 and 4 + 4 = 8 yielding an answer of 8; similarly, 2 × × = is treated as 2 × 2 = 4 and 4 × 4 = 16 yielding an answer of 16. These differences may be formalized in terms of how the internal registers are evaluated and used (see Mayer and Bayman, [8]). Table VI summarizes these strategies and shows that most subjects opted for ignoring the nonstandard sequence, but experts were far more likely to conceive of incrementing operations. Since incrementing is a feature of more sophisticated operating systems, this difference between experts and novices is sensible.

Similar results were obtained for sequences such as 2 × + 3 = and 2 + × 3 =. Most subjects ignored the first operator, yielding answers of 5 and 6, respectively. Some subjects reset the display, often yielding an answer of 0 for each problem. Some subjects used the incrementing strategy, for example, with answers of 7 and 12, respectively. There was also a subject who ignored the second operator, yielding answers of 6 and 5, respectively; and there was a subject who preferred multiplication to addition, yielding answers of 6 to both problems. The proportions of ignore, reset, and increment conceptions for novices and experts were quite similar to those shown in Table VI.

F. Nonstandard Sequences: Operator after Equals

Suppose the following key strokes were entered; 2 × = ×. Subjects who use the ignore conception of nonstandard sequences act as if this sequence is 2 ×, thus the answer is 2. Subjects who use the reset strategy give 0 as an answer. Subjects who use an increment strategy give answers such as 8, 16, or 4 depending on the particular kind of incrementing system and the subject's conception of when an expression is evaluated. Table VII summarizes these three strategies and shows that while most subjects rely on the ignore conception, a substantial minority of experts rely on increment strategies and a substantial minority of novices rely on a reset strategy.

Table VII. Three Major Conceptions of How to Evaluate Operation Following Equals.

Conception	Example: Problem	Example: Answer	Proportion of subjects: Novices	Proportion of subjects: Experts
Ignore the nonstandard sequence	2 × = ×	2	0.85	0.82
Reset the display	2 × = ×	0	0.15	0.00
Increment the display	2 × = ×	4 or 8 or 16	0.00	0.18

For 2 × 2 contingency table, $\chi^2 = 4.58$, df = 1, $p < 0.05$

G. Production Systems

One goal of the study was to formally describe the intuitions of each subject as a list of 13 productions, i.e., 13 condition-action pairs.[5] The left side of Table VIII (or IX) lists the 13 conditions that were present in the 88 problems we asked subjects to solve. For example, # after + means "pressing a number key after pressing an equals key" such as the last two keystrokes in the sequence 2 + 3. The preceding sections have summarized the different possible actions in general terms. The right side of Table VIII (or IX) gives the actions that may be associated with each condition. Actions are indicated as changes in the display (represented as D) or in the register (represented as R).

Table VIII represents one of our novices.[6] The subject evaluates only when an equals key is pressed (as indicated by P7), but does not evaluate an expression when a number key is pressed (as in P2 and P10) nor when an operation key is pressed (as in P4 and P11). The subject evaluates an arithmetic chain in the order from left-to-right; thus, for each action involving *eval* the procedure is left-to-right. The subject ignores all nonstandard sequences (as indicated in P5, P6, P8, P12, P13, P14, P15, P16). For example, on the problem 2 + 3 + 7 =, the subject begins by setting D = 2, R = 2. Then for the + key, P4 says that D = 2, and R = 2 +. For the 3 key, P2 says that D = 3 and R = 2 + 3. For +, P4 says D = 3, R = 2 + 3 +. For 7, P2 says that D = 7 and R = 2 + 3 + 7. Finally, when = is pressed, P7 says D = 12 and R = 12. As another example, consider the problem 7 + + =.

First, D = 7 and R = 7. Then when the first + is pressed, P4 results in D = 7 and R = 7 +. Then, when the second + is pressed, P5 results in no change, so D = 7 and R = 7 +. Finally, when = is pressed, P8 results in evaluation of 7 + which is 7; thus, D = 7 and R = 7.

Table IX represents one of our experts. The subject evaluates when an operation or an equals key is pressed (as in P4 and P11) rather than waiting for an equal key to be pressed (as in P7), but does not evaluate for pressing a number key (as in P2 and P10). The subject evaluates an arithmetic chain by performing multiplication before addition; thus for each action involving *eval* the procedure is multiply before add. The subject uses an incrementing procedure for most nonstandard sequences (as indicated in P5, P6, P8, P12, P13, P14, P15, P16). For example, on the problem 2 + 3 + 7 =, the subject begins by setting D = 2 and R = 2. Then for the + key, P4 says that D = 2 and R = 2 +. After 3 is pressed, P2 says D = 3, R = 2 + 3. After + is pressed, P4 says that D = 5 and R = 5 +. For 7, P2 yields D = 7, R = 5 + 7. Finally, when = is pressed P7 yields D = 12, R = 12. As another example, consider the problem 7 + + =.

First, we begin with D = 7 and R = 7. For the first +, P4 results in D = 7 and R = 7 +. For the second +, the number in the display (7) is added to the value in the register (7) to yield a value of 14 so D = 14 and R = 7. For the equals, the number in the display (14) is added to the number in the register (7) to yield 14 so D = 21 and R = 7.

V. General Summary and Recommendations

The results show that even though people are able to use their calculators to solve arithmetic problems, there are important individual differences in people's understanding of calculator language. In this paper, we have summarized differences in people's conceptions of when to evaluate an expression (immediately when a number key is pressed, when an operation key is pressed, or when an equals key is pressed), how to evaluate an arithmetic chain (left-to-right, multiplication-before-addition, etc.), and how to evaluate nonstandard sequences (ignore, reset, and increment).

In addition, there was a tendency for experts to differ from novices in the following ways. (1) Experts were more consistent than novices. (2) Experts tended to evaluate expressions when an operator key was pressed more than novices. (3) Experts tended to evaluate multiplication-before-addition in a chain more than novices. (4) Experts tended to increment the display for nonstandard sequences more than novices. Thus, the present paper provides some evidence that it is possible to describe user's concept of how calculator language works; in fact, in another paper [8] we provide production model representations for each subject. The fact that people have different conceptions of calculator operation, and that experts tend to develop more sophisticated ideas than novices, has implications for the design of calculator operating systems and instruction.

In a sense, this paper has been a plea for the use of cognitive objectives as well as behavioral objectives in users' learning of calculator languages. We need to be able to specify what we want the user to *know* about how the language works, as well as what we want the user to be able to *do*. The transaction approach provides a technique for describing the knowledge that a user currently possesses and the knowledge that we would like the user to acquire. One implication of this approach that warrants further study is that explicit training in the

[5] There are 13, rather than 16, productions because productions P1, P3, and P9 were never incorporated into the 88 test problems.

[6] Tables VIII and IX describe production systems for actual individual subjects rather than composites.

Table VIII. Production System for Subject N.

Production Number	Condition	Action	Description
P2	If # after +	then Set D=#, Set R="R+#"	Delayed evaluation and display
P4	If + after #	.then Set D=D, Set R=eval (R)+	Delayed display and immediately evaluated register
P5	If + after +	then Set D=D, Set R+=R+	No change in display or register
P6	If + after =	then Set D=D, Set R=R+	No change in display, plus added to register
P7	If = after #	then Set D=eval (R), Set R=eval (R)	Immediate evaluation and display
P8	If = after +	then Set D=eval (R), Set R=eval (R)	Immediate evaluation and display
P10	If # after ×	then Set D=#, R="R*#"	Delayed evaluation and display
P11	If × after #	then Set D=D, Set R=eval (R)*	Delayed display and immediately evaluated register
P12	If = after ×	then Set D=eval (R), Set R=eval (R)	Immediate evaluation and display
P13	If × after =	then Set D=D, Set R=R*	Delayed evaluation and display
P14	If × after ×	then Set D=D, Set R=R*	No change in display or register
P15	If + after ×	then Set D=D, Set R*=R+	Set register sign to add
P16	If × after +	then set D=D, Set R+=R*	Set register sign to multiply

Subject evaluates for equals sign only; subject ignores nonstandard sequences. Quote marks on the right side of an equality means that the entire expression is held in the register; eval (R) on the right side of an equals means that a single value is substituted for the expression previously in the register.

objects, locations, and operations (perhaps using a concrete model of the calculator) will enhance development of our cognitive objectives. The data presented in this paper are preliminary, but they provide clear evidence that people's knowledge can be described (based on simple prediction tests) and that there are large individual differences among users in what they "know" about calculator language.

The following recommendations are based on the idea that there should be as close a match as possible between the user's conception of how the calculator should operate and the actual operating system of the calculator. Each recommendation should be viewed as a tentative hypothesis that is subject to much future research, rather than as a fact that has been established through existing research.

(1) *Choose a calculator that corresponds to the intuitions of the user.* The most obvious recommendation is to choose a calculator that works the way that the user thinks a calculator should work, i.e., match the characteristics of the machine to the intuitions of the user. In our study of 33 novices and 33 experts, we found that Texas Instruments calculators gave answers that were most consistent with answers given by our subjects. However, there was disagreement between our subjects' answers and TI's answers on about 20 percent of the problems for both experts and novices. Rockwell and Sharp gave even poorer matches to our subjects' performance on the problems we used (see [8]). Far more study is required using more problems, different types of calculators, and more subjects before any definitive conclusions can be made concerning which calculators have the most intuitive operating system. Thus, it is beyond the scope of this paper to provide endorsements for specific calculators. However, the present study suggests that we cannot rely on choosing an "intuitive" calculator as the solution to all our problems, because even the best fitting calculator (i.e., in this case, TI) is considerably different in performance from what our subjects expect. Thus, there is need for instruction that helps produce user conceptions that are more consistent with the way calculators work.

Table IX. Production System for Subject E.

Production Number	Condition	Action	Description
P2	If # after +	then Set D=#, Set R="R+#"	Same as subject N
P4	If + after #	then Set D=eval (R), Set R=eval (R)	Immediate incrementing display
P5	If + after +	then Set D=eval, (D+R), Set R=R	Immediate incrementing display
P6	If + after =	then Set D=D, Set R=R+	Same as subject N
P7	If = after #	then Set D=eval (R), Set R=eval (R)	Same as subject N
P8	If = after +	then Set D=eval (D+R), Set R=eval (D+R)	Immediate incrementing display and register
P10	If # after ×	then Set D=#, Set R="R*#"	Same as subject N
P11	If × after #	then Set D=eval (R), Set R=eval (R)*	Immediate evaluation and display
P12	If = after ×	then Set D=eval (D*R), Set R=eval (D*R)	Immediate incrementing display and register
P13	If × after =	then Set D=eval (D*R), Set R=eval (D*R)	Immediate incrementing display and register
P14	If × after ×	then Set D=eval (D*R), Set R=eval (D*R)*	Immediate incrementing display and register
P15	If + after ×	then Set D=D, Set R+=R*	Same as subject N
P16	If × after +	then Set D=D, Set R*=R+	Same as subject N

Subject evaluates for operation or equals sign; subject increments for some nonstandard sequences.

(2) *Instruct users in the concepts underlying calculator language.* In particular, users should be able to relate each botton push to a series of transactions, i.e., to a description of events taking place in the display and registers. The transaction approach advocated earlier [6] for BASIC appears to apply equally well to calculator language. The locations should be made explicit and visible to the learner, perhaps by providing an erasable "scoreboard" for the display and internal registers. For each press, the learner should be able to alter the contents of the internal registers and display in accordance with the actual transactions.

(3) *Provide diagnostic tests and remediation based on the user's underlying concepts.* The present study has shown that although two users may be equally proficient at using their calculators to solve standard arithmetic problems, they may differ greatly in their conception of the calculator's operating system. Thus, a purely performance-based test of calculator skill does not tell a teacher what the user "knows". Instead, diagnostic tests should be carried out at the transaction level, for example, by asking the user to state what operations are applied to what objects at which locations in the calculator for each key press. Then remediation can be provided at the transaction level. For example, if a user indicates that the register is cleared to zero for any nonstandard sequence, this can be corrected by showing exactly what happens at a transaction level. Again, a concrete model (such as erasable display and registers scoreboard) could be used.

(4) *Challenge users to develop procedures for complex problems.* Once users have mastered the basics of calculator language and have developed appropriate conceptions of the underlying transactions, students should be encouraged to transfer their knowledge to more challenging problems. For example, a student who understands the transactions involved in "incrementing display" could be asked to figure a way to multiply a series of numbers, each by the same constant, or the user could be given a problem that involves a geometric progression, etc. Many exercise books are available [10, 14], but few provide any training on the principles underlying successful performance on creative problems.

(5) *Build on calculator language as a means to teach other languages such as* BASIC. Students have intuitions about how calculators work. This study has shown that the intuitions of any individual user are fairly consistent. A teacher can build on these intuitions and use them for transfer to programmable calculators, to BASIC and other languages.

(6) *Use the calculator as a starting point for the development of computer literacy.* Through interactions with calculators, the user may develop either a black box or a glass box approach to computers. It is important to start early in helping children (and adults) to see that calculators can be understood, for such an attitude is likely to transfer to other human/machine interactions. Mastery of the concepts underlying calculators is just a first step down the road to computer literacy.

Received 10/79; revised 1/81; accepted 2/81

References

1. DuBoulay, B., and O'Shea, T. *How to work the* LOGO *machine.* Dept. Artif. Intell., Paper No. 4, University of Edinburgh, 1976.
2. DuBoulay, B., O'Shea, T., and Monk, J. The black box inside the glass box: Presenting computing concepts to novices. Dept. Artif. Intell., Paper No. 133, University of Edinburgh, 1980.
3. Greeno, J.G. Cognitive objectives of instruction: Theory of knowledge for solving problems and answering questions. In *Cognition and Instruction.* D. Klahr, Ed., Erlbaum, Hillsdale, N.J., 1976.
4. Mayer, R.E. Different problem solving competencies established in learning computer programming with and without meaningful models. *J. Ed. Psych. 67,* 6 (Dec. 1975), 725–734.
5. Mayer, R.E. Some conditions of meaningful learning of computer programming: Advance organizers and subject control of frame sequencing. *J. Ed. Psych. 68,* 2 (April 1976), 143–150.
6. Mayer, R.E. A psychology of learning BASIC. *Comm. ACM 22,* 11 (Nov. 1979), 589–593.
7. Mayer, R.E. Elaboration techniques and advance organizers that affect technical learning. *J. Ed. Psych. 72,* 6 (Dec. 1980), 770–784.
8. Mayer, R.E., and Bayman, P. Analysis of user's intuitions concerning the operation of electronic calculators. Dept. Psych., Series in Learning and Cognition, Report No. 80-4, Santa Barbara 1980.
9. Mayer, R.E., and Bromage, B. Different recall protocols for technical text due to sequencing of advance organizers. *J. Ed. Psych. 72,* 2 (April 1980), 209–225.
10. Mullish, H. *The complete pocket calculator handbook.* Collier, New York, 1976.
11. National Council of Teachers of Mathematics. *An Agenda for Action: Recommendations for school Mathematics of the 1980s.* NCTM, Reston, VA, 1980.
12. Roberts, D.M. The impact of electronic calculators on educational performance. *Rev. Ed. Res. 50,* 1 (Spring 1980), 71–98.
13. Scandura, A.M., Lowerre, G.F., Veneski, J., and Scandura, J.M. Using electronic calculators with elementary children. *Ed. Tech. 16,* 8 (Aug. 1976), 14–18.
14. Suydam, M.N. State of the art review on calculators: Their use in education. Calculator Info. Center, Rept. No. 3, Columbus, Oh., 1980.
15. Wertheimer, M. *Productive Thinking.* Harper & Row, New York, 1959.
16. Young, R.M. The machine inside the machine: Implied models of pocket calculators. *J. Man–Machine Studies* (in press).