Are Mathematical Understanding and Algorithmic Performance Related?*

PEARLA NESHER

*Invited Address at the 1986 Annual Meeting of the American Educational Research Association, Division C-4, San Francisco, April 16-20, 1986

Algorithmic Performance and Understanding—Some Conceptual Clarifications

Posing the question concerning the relationship between mathematical understanding and algorithmic performance is, of course, simplistic and it is not my purpose here today to give a "yes" or "no" answer to it. What I propose to do in this presentation is to begin by examining the conceptual difference between the notions of "understanding" and "algorithmic performance" and then formulate a number of dilemmas that flow from the distinctions I will draw.

The term "algorithm" is derived from the name of the Arab mathematician Al-Khowarizmi who lived in the ninth century and formulated rules for the performance of the basic arithmetical operations we use to this very day. But, actually, the Greeks, Euclid and Erathostenes, as early as the 4th century B.C., elaborated algorithms that are being studied by children in our schools. An example of such algorithms is Euclid's method for finding the largest common divisor of two numbers; or the algorithm for finding prime numbers known as "the sieve of Erathosthenes."

It is, however, a mistake to think that an algorithm necessarily describes an *arithmetical* operation. Nowadays, with the widespread development and use of computers, the importance of algorithms goes far beyond the domain of mathematics itself. Instructions for how to operate a washing machine, or how to prepare a pancake can also serve as examples of algorithms. What, then, is an algorithm? Intuitively, we could say that it is a deterministic process for what is defined as an input set. To every member of the input set there is made to correspond by means of the algorithmic process one and only one member of the output set. Let us illustrate this with a non-mathematical example. Take the all-American pancake: its ingredients, such as eggs, flour, milk, sugar, salt and so on . . . will serve as our input whereas the instructions for performing the operations that will turn these ingredients into a pancake will be the algorithm. For those of you who are reminded of a computer, let me add that the bowl, the mixer and the oven would be the counterparts of the hardware.

This algorithm, like others, is characterized by being finite and by being performed step by step in a prescribed order. Each step will be performed on the basis of the current state within the process. (For example, the frying on the griddle will take place only after the ingredients have been mixed; the mixing will occur in a given order; the proper quantities of the ingredients having been measured according to the instructions in the recipe). It is typical of an algorithm that if we define the process precisely enough, we will always reach a pre-determined outcome: in our case it will be a pancake. Clearly, the recipe must be more detailed and include such information as the temperature of the griddle, frying time, and so on. Clearly, the amount of detail included in the description of an algorithm's basic operations will be a function of the knowledge level of its user. Our pancake instructions will vary greatly whether we formulate them for a chef or for a robot-cook trusted with the formidable task of preparing pancakes with the help of the computer to which our algorithm has been fed. Whether the process is long or short, it will always be final; and since it is a deterministic process, whenever we operate on the proper input we will get the desired outcome. There are other specifications we may need as we go along, but let us now pause and ask "What do we mean by *understanding*?"

Everything I have done so far in talking about the notion of an "algorithm" was designed to help you reach an *understanding* of that notion. But, clearly, I have not performed an algorithm in so doing, and I cannot therefore be certain of the outcome of my efforts. Since I cannot assume that each one of you has indeed understood what an algorithm is, let me examine more closely what I actually did in trying to explicate this concept to you.

My move was to try and relate the term I wanted you to understand to some other knowledge system that I assumed was already known and familiar to you. Thus, for example, I assumed that each of you knows what I am talking about in discussing the preparation of pancakes (and did not choose to speak about "falafel preparation" which would be a better example in my own country). I assumed that you are familiar with the ingredients which appear in the pancake recipe, that you have a sense of the final texture of the mix and even an idea of what the finished product should look like.

My explication involved other assumptions as well, e.g. that everyone has some experience of failed attempts at some food preparation and will know how to appreciate the importance of maintaining the proper order in performing the required operations. In other words, I tried to give meaning to the concept, to the word "algorithm" by appealing to an already existing system of concepts. I say

For the Learning of Mathematics 6, 3 (November 1986)
FLM Publishing Association, Montreal, Quebec, Canada

"system" because my explication was not a mere search for a synonym but, rather, the description of a system comprising several elements which are interrelated in some describable way, whether it is a relationship defined by ordering or in some other way. This is not the only way to define new, unknown terms, and the notion of understanding, as I shall indicate later, is not confined to single terms or concepts.

Having given you a sense of what I mean by understanding, let me emphasize what understanding is, or rather is not, by contrasting it to algorithmic performance. Understanding has several levels to it which are not always ordered. Understanding is never finite and complete, rather it is open ended. We do not know precisely enough what the states that lead to understanding are, and we cannot acquire understanding in a mechanistic manner that will assure the pre-determined outcomes. Otherwise all our instructional problems would be rendered trivial.

Let me now go back to our discussion of algorithms. Why are algorithms considered so important? There is a dream that algorithmic performance could be fully handled in the future by computers and thereby liberate human beings from many tedious activities. This is probably true. Computers can perform algorithmic procedures in a marvelously precise and fast way. Computers, however, cannot perform even the simplest task if it is not defined for them by means of a precise algorithm. Unlike computers, human beings are capable of performing operations which they cannot yet "translate" into algorithmic form.

As is well-known, researchers in the field of Artificial Intelligence attempt to construct computers whose operation will resemble as closely as possible the operation of natural, human intelligence. In 1950 Turing proposed a simple test which will help determine if a computer is indeed as intelligent as a person. The test runs as follows: in one room there sits a person X who is connected only through a computer terminal to two other rooms. In one of these rooms there sits another person Y and in the second is placed a computer C. Now X is given the task of finding out in one hour through questioning which of the two is the person and which the computer. If the computer can pretend to be a person for one whole hour, then it has won and must be considered to be as intelligent as a human being. According to the rules of the game, the computer can lie and respond to the question: "Are you a computer?" by saying "No, I am the laboratory assistant here."

Needless to say, to this day no computer, however sophisticated, has passed Turing's test. As the philosopher Hilary Putman [1981] comments, if the computer is asked to answer the following question: "Please open the window and tell me what plants are growing in the garden," could it really do so? Certainly not, and not only because of the motor element involved but simply because of its incapacity to distinguish between green grass and an apple tree. Or in the words of David Harel [1985]: whereas we cannot but marvel at the computer's capacity to process data from dozens of X-ray pictures of the brain and turn it into a picture which gives an intersection of the brain and helps to discover tumors in it, we must still note that the computer cannot analyze a regular photograph of the same patient and determine whether he is young or old, a feat any ten-year-old can readily perform.

This example illustrates the difference between the capacity to perform many complicated and subtle operations at high speed and with great skill (but following a given algorithm) and the ability to perform successfully within a framework of intuitively apprehended rules the route to which cannot be clearly specified. This ability is part of our special human endowment.

But let me state as clearly as I can: I do not wish to claim that machines perform in an algorithmic fashion whereas human beings operate according to intuitive mental rules. This is certainly not a valid distinction: human beings perform innumerable operations which are algorithmic in nature. We encounter such operations whenever we think of habits or automatic actions. Moreover, human actions become habitual or automatic just because they involve an unchanging algorithm and many exact repetitions that rendered it automatic or habitual. Such are the actions of opening the door by turning the door handle, driving a car, or signing a form. In all these examples we perform automatically and will fail to do so only when our activity is obstructed—when, for example, we come upon a door we are not sure whether we should push or pull. At this moment our automatic performance will disappear and we will have to pause and make judgements based on the written sign on the door, or on observations of other people's conduct, or on trial-and-error pushing or pulling attempts.

The importance of the automatic-algorithmic operation in human conduct lies in its efficiency. Since it is an automatic habit, every step in the process is performed instinctively on the basis of a previously defined state of affairs without further intervention of control and judgment processes. This is efficient as it liberates human thought to deal with context and matters that do not permit the achievement of full automation. We are thus enabled to drive while conducting a conversation with the person sitting next to us. The driving operations are instinctive and automatic and we are mentally free to follow and assess our partner's newly formulated argument, for example.

An important question suggests itself in this context: is the above account merely describing a temporary state of affairs, one that will hold until all the operations humans can perform are transferred to computerized algorithmic procedures? It is a complex question and I hope by the end of this talk to have convinced you that no such danger lies ahead of us, especially not in the area of mathematical learning.

As noted, automatic processes come to a halt when something "goes wrong." What does "goes wrong" mean in our context? It means that we have reached a place in our performance that we never expected. Here again, we note the difference between human beings and machines: a computer will stop when it fails to read the syntax we use to speak to it (though some sophisticated computers can even track syntactic errors and correct them). But this does not apply when the error is due to a thinking error of the person

who wrote the algorithm for the computer If there are no syntactic errors the computer will perform the algorithm from beginning to end. Say, for example, that we wanted to count the number of sentences in which the word "understanding" appears in sentence-final position and the algorithm for finding the end of sentences was incorrectly formulated. As a result we got the number of times the word "understanding" occurred in the text as a whole Errors in thinking are usually more difficult to detect. Needless to say, the computer cannot detect them at all and a human being will "catch" them only when the distance between the expected answer and the one received is considerable. For instance, in the previous example, if the text we gave the computer to check for occurrences of the word "understanding" is short and this word occurs in it many times not only in sentence final position — we are likely to detect the error easily. But this will not happen if the text is long and we have no idea about the number of occurrences we can expect. Compare this to the situation of the person who arrives in a foreign country and is not familiar with its currency. Inquiring about the cost of a hotel room, he or she will not know whether the price cited is high or low. Consciousness of and the expectation for a particular outcome is, in fact, the basis of the control we have over the results of our actions.

This brings me to a broader definition of the process of understanding than the one I have given before: understanding is the control we have over the process of knowing [Dan Nesher, 1986]. This control is found at various points in the knowledge process including steps in the learning of the algorithm itself and certainly in its various applications. We may, therefore, speak of different levels of understanding, as I will presently demonstrate

In learning algorithmic performance does one have to understand?

(The Case of Counting)

Let me now discuss several studies related to mathematics learning which are relevant to the issues of understanding and algorithmic performance. One of the first explicit acts of learning in early mathematics is the acquisition of counting. Counting is a well know algorithm which if followed accurately will lead to the knowledge of the number of any given set of objects. For example, if I want to know the number of apples in a bag, all I have to do is count them. This is usually done by touching (or pointing at) each of them separately and, while doing so, saying an ordered list of number names If I start from 1 and I mention each number name without omitting any of them, then the number name that I mention when touching the last apple will be the counting number of the entire set. And if I try to count the same apples again and this time reach the number 7 instead of 8, for example, I, as an expert in counting, will know that I have made a mistake. The knowledge that it is impossible to reach two different numbers in counting the same pile of apples is part of the control system that each expert in counting has.

Let us, however, watch a two or three year old child who starts to imitate his mother and say "one, two, three . . ." What does he understand by this? Does he know that this is an algorithm that will lead him to the knowledge of the counting number of any given set? Of course not. In order to appreciate this act as an algorithm for reaching the counting-number of a set the child has first to master the full algorithm and not merely mention some number names

The process of learning all the sub-skills needed for the counting algorithm takes about three years for the young child. As everyone who has read the very detailed studies in this field now realizes, the child not only has to learn each of a number of sub-skills which requires its own understanding, but he also has to be able to integrate them The child has, for instance, to have a phonetic understanding about the order of the sounds that he listens to; these will later on be segmented into distinct number names serving to tag objects. The objects themselves have to be distinct and each of them has to be touched just once, etc., etc. My claim is that perhaps the three year old child does not have the full understanding of the role of each sub-skill in the counting algorithm, but he must have some understanding of how to execute each of the above sub-skills. He has to understand what it means to touch just once; or that the order among the number-names is significant, etc.

I would like to emphasize that *understanding* here is interpreted by me as the control system the child has about the performance of each sub-skill. If, for instance, the child knows what sound has to come after another, performs it instinctively, and notices when it goes wrong, in my view he has acquired a control system for this sub-skill. This is, of course, not the full understanding of the counting algorithm, but this is a partial understanding that is necessary for performing each of the sub-skills, which is in turn a necessary step before one can master the entire complex process of counting correctly.

No wonder, then, that research on this performance by Gelman, Fuson, Riley and others explicitly connects these sub-skills to principles. They name them as the counting principles that form the basis for a competence theory that underlies the performance of counting. Acquiring the principles is equated here by me as *understanding*. One has, however, to be careful. Along the way, for about three years the child does not have the same understanding of the counting algorithm as the expert has. The child does not understand that this algorithm will lead him in a reliable manner to the knowledge of the number of any given set of objects; yet he understands some principles underlying the sub-skills of counting.

At this point I would like to present a dilemma. Maybe the child cannot achieve the full understanding of counting as a reliable algorithm unless he has acquired a sufficient mastery in the sub-skills, which are by themselves algorithmic in nature, so as to be able to reflect on them and find out their invariants. In the case of counting, perhaps, unless the child is skilled enough to mention the number-names without errors, and able to touch each object just once, he will not be able to reach the understanding of counting as a reliable algorithm. Only when he reaches this kind of reflection on the invariants of counting will he have

developed a control system, which in my view, would constitute *understanding*. Piaget gives an example which is pertinent to this issue:

> He (a boy of 4 or 5) was seated on the ground in his garden and he was counting pebbles. Now, to count these pebbles he put them in a row and he counted them one, two, three . . . up to ten. Then he finished counting them and started to count them in the other direction. He began at the end and once again he found he had ten. He found this marvelous . . . so he put them in a circle and counted them that way and found ten once again. [Piaget, 1964, p. 120]

I would like to suggest that this was the moment of insight which led the boy to full understanding.

What I have tried to emphasize is that perhaps the separation between algorithmic performance and understanding is impossible at any stage of the learning. Only after the learning has been completed and the performance become instinctive can one speak about mechanically efficient performance that does not activate thinking and understanding processes. Yet, when some failure occurs, all the bells begin to ring and the control system is called into action to re-establish regularity of performance

Studies dealing explicitly with the relation between algorithmic performance and understanding

It seems, then, that in learning a certain algorithm one needs a prior understanding of the sub-skills contributing to the algorithm which later serve as a control system for the entire algorithm. To my surprise, research dealing directly with the question of the relation between algorithmic performance and understanding does not support this thesis.

I would like to report on two such studies. The first one is a study by Bilha Zuker, my former M.A. student in Israel. Her study was about algorithmic performance and understanding in decimals. Her main hypothesis was that students who understand better will also perform better in decimal calculations. For that purpose she composed two separate tests: one test for examining the student's algorithmic performance and the other for testing understanding. The test for algorithmic performance included the four simple operations with decimals (addition, subtraction, multiplication and division: see Appendix 1); whereas the test for understanding included items that called for reading and writing decimals, comparing decimals, the concept of density in decimals and the ability to predict the outcomes of multiplication and division with decimals (see Appendix 2). The reliability of both tests was examined and found to be 0.82 for the first one and 0.93 for the second (Cronbach Alpha). The tests were given to 240 students in grades 7, 8 and 9 (the ages of 13 to 16), both honor and regular students. The analysis considered also whether the students' aptitude was high or low since many of us believe that good students are better because they understand more than the weak students.

The findings of this study show that there is no correlation at all between the two tests for the entire population. There was a correlation of 0.27 for the high level students (which was significant because of the big N) and no correlation at all for the low achievers (despite the big N). Indeed the low achievers received lower scores than the high achievers on both the algorithmic and the understanding tests, yet no relationship was found between the two tests. The next step was to find out possible relations between certain subsets of the two tests. Of particular interest was the comparison between the subtests that asked for the algorithmic multiplication and division of decimals and the part of the understanding test which asked for predictions whether the outcome of multiplication or division in decimals would be greater, smaller or the same as the number given the student to operate on. Again, results were the same. Even for the high level students the correlation was 0.25 and for the low level students no correlation was found.

All this did not persuade my student to abandon her hypothesis. One should remember that the main motivation for her research was to "prove" how important it is to teach toward understanding and she could not believe it was not true, neither for high achievers nor for low achievers. So she looked more carefully into the patterns of gaps between the scores of the two tests. She found then that the split on individual levels between the two tests is often more than 15 points on a scale of one hundred in both directions (i.e. either higher on the algorithmic test, or on the understanding test). As can be seen in Appendix 3, this happened in both populations. She also found that the distribution between the two populations on this split was almost the same (see Appendix 3).

Disappointing, is it not?

At approximately the same time, however, another study was carried out in Pittsburgh by Resnick, Omanson and Peled which dealt with understanding place value concepts and performance on the subtraction algorithm. In this study, again, a direct confrontation between performance according to syntactical rules vs. understanding underlying principles was made. The essence of the study was the comparison of two groups which were different in the kind of instruction they received: one of them received mapping instruction whereas the other received prohibition instruction.

The mapping instruction required the child to perform the same problems in blocks and in writing, alternating steps between the two. This method was designed to teach children the semantics of the procedure, i.e. *understanding*. The prohibition instruction was different. It worked directly on children's "buggy" algorithmic performance. In this case the experimenter began the prohibition instruction by introducing herself as the student's subtraction robot who would do problems for the student but who needed explicit directions about what to write. If the student, then, said something wrong the experimenter said: "I am not programmed to do it that way, try again." As Resnick writes:

> The central question we have been considering is the relationship between *understanding* of basic princi-

ples of numeration and *performance* of arithmetic procedures. It is clear that buggy performance involves violation of basic principles, and thus it seemed reasonable to suppose that if children acquire the principles they would be less likely to engage in buggy performance. [Resnick, 1984, p. 11]

The findings of this study were also surprising. The findings show that for written calculations neither group improved reliably between the pretest and the second posttest. Although there was a temporary improvement for many children, old bugs remained or reappeared at the second posttest, or new ones were invented. The researchers expected such an outcome for the prohibition group because of their theory at the beginning of the experiment that "only a command of the semantic principles of subtraction would successfully block the use or construction of buggy procedures that violate those principles". [Resnick, 1984, p. 10]

Though the Resnick, Omanson, and Peled study is presented here very briefly one should note that it was not a single experiment, but rather a series of experiments aimed at dealing with differences in initial knowledge; differences in amount of instruction and in amount of manipulating of blocks during instruction; differences in mastery during instruction; verbalization of quantities during instruction, etc. Yet the study found that children who received mapping instruction did not improve significantly on their calculation performance, and as Resnick says: "Quite obviously, the children did not always call upon all of their relevant knowledge when doing calculations." [1985, p. 54]

We have, then, two basic and well controlled studies that were specifically designed to raise the question of the possible contribution of *understanding* to algorithmic performance. The two studies differ in the way they address the same question. Zuker's study deals with an overall control system that one has for a given field vs. the ability to perform algorithmic operations in the same field. The second study tries to examine the impact of the infusion of semantic and understanding principles into the instruction of a certain algorithm. Both failed to prove their hypotheses. No one has succeeded in demonstrating that *understanding* improves algorithmic performance. Though we all feel, intuitively, that this is the case, we are still in a state of wishful thinking without grounded facts. From now on all is mere speculation.

In order to save the theory that says that understanding will improve algorithmic performance, Resnick suggests two hypotheses: the first hypothesis has to do with how children *represent* to themselves the substraction problem: whether it is for them a matter of manipulating symbol strings (which is syntactic in nature), or whether it is for them a matter of dealing with quantities (which is semantic in nature). The second hypothesis has to do with the initial learning of the algorithm vs. correcting an already acquired buggy procedure, a point to which I will refer later.

Resnick suggests that one of the goals of the initial learning of a procedure is "an effort to help children derive procedures from the principles of the procedural domain that might prevent buggy rules from ever becoming automated". [Ibid, 1985, p. 56] Resnick also tries to draw a distinction between *automatic* calculation routines and learning principles. She suggests that monitoring and reflection is contradictory to the very notion of automaticity, and if error-free calculation is to be achieved, attention to appropriate forms of automated—that is, nonreflective practice—will be needed along with attention to the building of understanding.

Such a proposal assumes the distinction between two kinds of learning, namely, learning automatic skills and learning for understanding, each of which has to be learned in a different way.

Let me clarify the last claim with the help of a nonmathematical example. Many of us will agree that learning to drive a car is not the same as learning the mechanics of the car. These are two distinct learnings. When we intend to learn how to drive, we will do it by means of drill and practice until we are able to perform the entire algorithm in an automated manner. Driving on a busy highway demands quick automated reactions, therefore we will aim at automated instinctive performance.

This is not the case in regard to the learning of the car's mechanics. Knowing the mechanics of a car is needed for one who intends to repair his car. In this case, the diagnosis of the fault to be repaired is based on the knowledge of the car's mechanics but is reached by a sequence of reflections and deliberations on the nature of the fault and its possible causes.

Are we now in a position to decide what kind of knowledge we would like to teach?

In the case of cars, modern society prepares many more drivers than mechanics. Is this also the appropriate conclusion for mathematics learning at school? I doubt it. If we reach the conclusion that learning mathematical algorithms and learning toward understanding are two distinct types of learning, there is a real question whether, in our modern era when small calculators are so handy and generally available, anyone would wish the young child to perform algorithms which can only be attained by long and tedious procedures and which are doomed to failure merely because of their length and the probability of making errors at each step (and I mean, here, long multiplications, long divisions, etc.).

Why don't we teach for understanding and leave the algorithms to computers?

In the first section I have already stressed that the computer can perform algorithms better than human beings (probably just because it lacks so many human qualities). On the other hand, the computer cannot write algorithms for itself since this requires knowledge and understanding that the computer lacks. Therefore one might propose that instead of training human beings to become efficient machines that can perform various mathematical algorithms in an automatic way, we should leave these for the computer and teach our students only topics which require and enhance mathematical understanding, such as problem solving, so that they can write more and better algorithms for the computer in the future

And this is again a dilemma

Let me consider this proposal by presenting another aspect of the problem Our group in Israel has tried to prepare a curriculum unit that will help the students "understand" the concept of an average (or mean) so that they should not just add numbers and divide them mechanically in a simple, algorithmic fashion. Given our conception of "understanding" as a control system we wanted the students to understand that the mean is a number that represents a set of numbers; we wanted him to know that it does not need to be one of the members of the set; we wanted him to know that this number (the mean) cannot be outside the range of numbers we have given, and so on.

Some exploratory instructional efforts very quickly made us realize that one gets nowhere in discussing all these characteristics before the students have some minimal experience with "doing averages" (or means). It turned out that only after the students have performed a certain number of exercises on means (averages) does it begin to make sense to have them reflect on what they have done with the purpose of reaching some understanding of properties of average numbers (or means).

We were therefore forced to make changes in the unit we had written and start it with the performance of a new algorithm (which we labeled "the mean (average) algorithm") and thus provide the students with an object for subsequent reflection. Does this imply that initial learning is mechanical in nature? My answer is negative. In the learning of the mechanical aspects of "doing means," too, there were elements of understanding. For example, one must know how to determine the number to be divided in, in computing the average number, as it does not necessarily appear in the formulation of the problem; one must know what happens when one of the numbers is 0, and so on. In short, in order to respond to various inputs one must understand the constraints of the algorithm. These present themselves only when a sufficiently great number of attempts at operating with it has been made. All these are elements of understanding that are essential to proper algorithmic functioning.

We observe, then, different levels of understanding. In the first phase we have an understanding that may be termed syntactic or practical. This is an understanding of the constraints which operate on the new algorithm which creates a new number to represent a set of numbers. And we have a different kind of understanding—a semantic understanding—of the concept of an average. We could not speak about this concept before it was demonstrated in the form of a certain procedure. Perhaps in the era of calculators and computers we do not need to practice this algorithm until it becomes automatic knowledge, but it is hard to imagine any talk and explanations concerning the characteristics of averages (means) without first hand experience with the "average algorithm" itself.

The story, however, does not end there. How is the concept of "mean" understood by someone who has learned about "standard deviation" and "variance" as compared to someone who has not?

There are, therefore, different levels of understanding, all of which occur before the algorithmic can become automatic. It appears that the question as to whether teaching should be designed towards algorithmic performance or towards understanding is a complicated one and does not admit of a clearcut yes or no answer. I believe the example I gave at the beginning of my talk of the complexity of the counting operation is not unusual in mathematical learning. It seems to me that all mathematical procedures require different levels of understanding, just like counting and the computing of averages. But even in such cases we must take care not to confine our teaching to syntactics alone, because in our day this kind of knowledge had really better be left to computers.

Perhaps we should conceive of the learning of the long subtraction algorithm that has attracted so much research effort in recent years as learning designed to deepen the understanding of place value rather than to replace the calculator, just as the learning of standard deviation serves to deepen the understanding of the concept of mean (average).

Perhaps this holds for the case of the car, too: the first phase of the driving algorithm may require understanding at the level of practical constraints—the difference between the clutch and the brakes and what each does; the relationship between the speed and the gears and how the latter are to be changed; how one drives backwards and how the lights are to be turned on. All these are elements of understanding that form part of the learning process. Only later, following much practice in braking and fitting the steering-wheel movement to the shape of a bend in the road does the operation become automatic and instinctive, and then all the elements of understanding that were needed for the initial performance become action rules at a lower level of consciousness.

The following example, taken from John Anderson, will illustrate it best:

> A number of years ago when my wife taught me how to use a stick shift [in their car], one of the questions I asked her was whether I should take my foot off the gas when shifting gears She said that I should keep my foot on the gas. But we did not like the results. So she took the driver's seat and we both watched what she did when she shifted—she did take her foot off the gas [Anderson, 1980, p. 225]

Are we interested in such an automatic performance of mathematical operations?

I do not think so. We have calculators and computers that can perform any algorithm that we can define, accurately and quickly. But rejecting the need for a type of learning that would lead to automatic instinctive performance does not imply a rejection of the learning of the algorithm itself. Perhaps in some cases the learning of the algorithm is noting but the procedural learning required for the learning of a new mathematical concept that could not be learned in any other way.

Let me conclude.

I have started my presentation with an attempt to clarify the distinction between algorithmic performance and understanding. I have emphasized that one of the advan-

tages of mechanical performance that grows out of the re-iteration of algorithms is the liberation of thought It is also obvious that such mechanical procedures can and should be executed by machines. In the remainder of my presentation I tried to argue that as far as mathematics learning is concerned, the above simplistic dichotomy between man and machine, presented as a dichotomy between learning algorithms and understanding, is a superficial and misleading dichotomy. My argument was based on two considerations:

> *First,* so far, no research on mathematical performance has succeeded in supporting a clear-cut hypothesis concerning the relationship between success in algorithmic performance vs. success in understanding; nor is there any direct evidence about the contribution of understanding to algorithmic performance. *Secondly,* I have questioned the possibility of teaching for understanding in mathematics without attending to the algorithmic and procedural aspects.

The shared belief among math educators so far is that one should teach for understanding since this contributes to developing the monitoring control system that the student needs in doing algorithms. One of my main points is that one should look at the same relationship from the opposite angle, namely, that knowing the algorithms and the procedures contributes to the student's understanding.

If it is true that algorithms have liberated us from the need to think and that, on the other hand, algorithms are needed to teach us to think about objects which are themselves algorithmic in nature, what remains for us to do is to consider which algorithms we will want to use to free ourselves from thinking, and which can be best used in order to further our thinking and understanding. Or, if you will, what kind of division of labor we wish to establish between ourselves and our computers in the years to come.

Bibliography

Anderson, J R. [1980] *Cognitive psychology and its implications* San Francisco: W H. Freeman and Company

Fuson, K C. [1981] An analysis of counting-on solution procedures in addition. In Carpenter, T., Moser, J. and Romberg, T. (Eds.) *Addition and subtraction. a developmental perspective* Hillsdale, NJ: Erlbaum

Fuson, K C. & Hall, J W. [1983] The acquisition of early number meanings. In H. Ginsburg (Ed) *The development of mathematical thinking* New York: Academic Press

Gelman, Rochel [1978] Counting in preschoolers: what does and does not develop. In Siegler. R S. (Ed.) *Children's thinking. what develops* Hillsdale, NJ: Erlbaum

Gelman, Rochel & Gallistel, C.R [1978] *The child's understanding of number.* Cambridge, Mass: Harvard University Press

Greeno, J G. & Simon, H.A. [1984] Problem solving and reasoning. Technical Report No. UPITT/LRDC/ONR/APS-14

Harel, David [1985] *Fundamental topics in computer science.* Ministry of Defence and Tel Aviv University, Israel

Inhelder. B. & Piaget, J [1969] The early growth of logic in the child W W Norton & Company, New York (First published by Routledge & Kegan Paul Ltd , 1964)

Nesher, Dan [1986] The understanding of "understanding" : a pragmatist approach (manuscript)

Nesher, Pearla [1986] Learning mathematics: a cognitive perspective *The American Psychologist* (forthcoming)

Nesher, Pearla [1984] Precursors of number in children: a linguistic perspective. A paper presented at the second Tel-Aviv Annual Workshop on Human Development, October 1984, Tel-Aviv

Piaget, J. [1952] *The child's conception of number.* W.W Norton Inc. (first published in French, 1941)

Piaget, J. [1967] *Biology and knowledge.* Chicago: University of Chicago Press

Resnick, L.B. & Omanson, S F. (in press). Learning to understand arithmetic. In R. Glaser (Ed.) *Advances in instructional psychology* (Vol 3). Hillsdale, NJ: Erlbaum

Resnick, L.B. [1984] Beyond error analysis: the role of understanding in elementary school arithmetic. In H N Cheek (Ed) *Diagnostic and prescriptive mathematics: issues, ideas and insights.* Kent, OH: Research Council for Diagnostic and Prescriptive Mathematics (1984 Research Monograph)

Zucker, B. [1984] The relation between understanding and algorithmic knowledge in decimals M A. Thesis, University of Haifa

Appendix 1

DECIMAL TEST (X)

Copy the following exercises vertically and solve:

Addition:

1) 56.1 + 1.083 =
2) 9.2 + 4153.015 =
3) 512 + 4.56 =
4) 2.9 + 0.326 + 137 =

Subtraction:

1) 18.2 – 1.82 =
2) 5 – 0.075 =
3) 0.1 – 0.0983 =
4) 1000 – 9.99 =
5) 76 57 – 14 =

Multiplication:

1) 6.09 × 7.5 2) 12.6 × 40.9 3) 9.47 × 500

Division:

1) $\overline{202.5}$ /45 2) $\overline{253}$ /40

3) $\overline{3.892}$ /0.28 4) $\overline{258.3}$ /126

5) $\overline{13.87}$ /0.073

Appendix 2

TEST FOR UNDERSTANDING (Y)

A.

1) Circle the larger number in each pair of numbers. If you think they are equal, mark =.

1	4.63	4.8	16	6.54	6.180
2	3.47	3.2	17	2.09	2.3754
3	4.08	4.7	18.	1.23	1.540
4.	0.100	0.25	19.	13.6	13.21
5.	4.733	4.08	20.	12.536	12.25
6.	7.85	7.850	21.	0.092	0.6
7	2.35	2.305	22.	2.35	2.350
8	4.4502	4.45	23	2.85	2.085
9.	2.35	2.035	24	0.370	0.25

10.	0.21	0.130	25.	0.36	0.5
11.	0.06	0.428	26.	3.621	3.62
12.	0.470	0.32	27.	4.089	4.23
13.	0.231	0.32	28.	13.680	13.73
14.	0.89	0.6	29.	17.037	17.5372
15.	0.06879	0.621	30.	0.450	0.4

B.

1) How many numbers are between 4.6 & 4.7

2) Write down a number which is between 1.29 & 1.3

C. In the following, do not compute Decide whether the result is bigger, smaller or equal to 328 and mark the correct answer

1. $328 \times 0.685 =$	bigger	smaller	equal
2. $328 \times 1.0 =$	bigger	smaller	equal
3. $328 \times 1.249 =$	bigger	smaller	equal
4. $456 : 0.289 =$	bigger	smaller	equal
5. $456 : 1.00 =$	bigger	smaller	equal
6. $456 : 1.352 =$	bigger	smaller	equal

D Read each number and write it down in words *without* using the word "point"

1. 0.39 _____
2. 2.5064 _____
3. 0.081 _____
4. 1,608.003 _____
5. 0.9010 _____

E. Write down the following numbers

1. 53 tenths
2. 6 tenths and 2 hundredths
3. 5 units and 6 hundredths
4. 5647 thousandths
6. 429 tenths

Appendix 3

THE DISTRIBUTION (IN %) OF THE DIFFERENCE BETWEEN X (ALGORITHMS) AND Y (UNDERSTANDING)

The Difference X-Y	Honor Students	Regular Students
+15	19%	24%
+5	17%	10%
0	20%	21%
-5	17%	17%
-15	27%	28%

$X > Y$ 35%
$Y > X$ 45%

Rather than as a positive element, I am inclined to view the Greek efforts to formulate and prove knowledge, originally attained in a simple and straightforward way, by means of clumsy methods and governed by strict conventions, as a symptom of a terrifying dogmatism, cherished by mathematicians and which, from Greek mathematics onwards to the present day, has hampered and sometimes gravely endangered the dissemination of mathematical knowledge and method.

Hans Freudenthal