DOCUMENT RESUME

ED 271 092 IR 012 120

AUTHOR Salomon, Gavriel; Perkins, D. N.
TITLE Transfer of Cognitive Skills from Programming: When and How?
SPONS AGENCY John and Mary R. Markle Foundation, New York, N.Y.; National Inst. of Education (ED), Washington, DC.
PUB DATE [85]
GRANT NIE-G-83-0028
NOTE 20p.
PUB TYPE Information Analyses (070) -- Viewpoints (120)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Cognitive Ability; *Learning Processes; Literature Reviews; Problem Solving; *Programing; Programing Languages; *Skill Development; *Teaching Methods; *Transfer of Training
IDENTIFIERS Automatization; LOGO Programing Language

ABSTRACT

Arguing that the most widespread argument in favor of the teaching of programming concerns its possible impact on generalizable cognitive skills, this paper addresses the "how" of transfer. The outlines of a theory of the mechanisms of transfer are presented, the theory is used to examine the contrasts between certain studies that did and did not obtain positive transfer results from programming, and a discussion is presented of the kinds of transfer that can be expected from programming and when. Two roads to transfer are identified (high and low) and examples of each are provided. In addition, six broad categories of transfer that might occur with programming are described: (1) mathematical and geometric concepts and principles; (2) problem solving, problem finding and problem management; (3) abilities of formal reasoning and representation; (4) models of knowledge, thinking, and learning; (5) cognitive styles; and (6) enthusiasms and tolerances. It is concluded that although programming instruction can improve cognitive skills under the right conditions, implementing such conditions on a wide scale may be difficult, and programming will have to compete in the intellectual and economic markets with a number of other approaches to the same general problem. A five-page list of references is provided. (JB)

Transfer of Cognitive Skills from Programming: When and How?

Gavriel Salomon

University of Tel Aviv

D. N. Perkins

Harvard Graduate School of Education

To appear in: Journal of Educational Computing Research

Authorship is alphabetic. The authors gratefully acknowledge support from the John and Mary R. Markle Foundation to Gavriel Salomon, from National Institute of Education grant number NIE-G-83-0028, Learning to Reason, to David Perkins, and from the NIE-sponsored Educational Technology Center at the Harvard Graduate School of Education. The views expressed here do not necessarily reflect the position or policy of the supporting agency.

Abstract

Investigations of the impact of programming instruction on cognitive skills have yielded a few positive and many negative findings. To interpret the mixed results, we describe two distinct mechanisms of transfer -- "low road" transfer, resulting from extensive practice and automatization, and "high road" transfer, resulting from mindful generalization. High road transfer seems implicated where positive impacts of programming have been found; insufficient practice and little provocation of mindful abstraction are characteristic of investigations not demonstrating transfer. Our discussion affirms that programming instruction can improve cognitive skills under the right conditions, but cautions that implementing such conditions on a wide scale may be difficult and that programming instruction must compete with other means of improving cognitive skills.

Transfer of Cognitive Skills from Programming: When and How?

Teaching programming in schools may have many justifications. It could be argued that programming is a necessary component of computer literacy, a skill that increases employment possibilities, and a competency that liberates one from dependency on prepackaged programs. But perhaps the most widespread -- and the most persuasive if supported -- argument in favor of the teaching of programming concerns its possible impact on generalizable cognitive skills.

For example, Feurzeig, Horwitz, and Nickerson (1981) argued that programming provides an opportunity to develop rigorous thinking, learn the use of heuristics, nourish self-consciousness about the process of problem solving, and in general achieve significant cognitive advances. Similarly, Linn (1985) analyzed the cognitive requirements of different levels of programming, such as precision and structural organization, expecting them to be potential cultivators of the processes involved in general problem solving. Papert (1980) urged that programming experiences in Logo could equip the learner with powerful ideas about knowledge and learning in general. But the research in the field provides only partial and often conflicting evidence to support these hopes (for reviews, see Blume, 1984; Land & Turner, 1985; Pea & Kurland, 1984a). Even if worthwhile transfer can happen, it occurs only in some cases but not in others. Somehow, an activity supposed to be mind stretching and thought provoking yields elusive and unsystematic results.

Some help in making sense of the circumstances comes from distinguishing between <u>what</u> might transfer from programming and <u>how</u> transfer occurs. While many have addressed the former, little is known about the latter. For instance, recognizing that the procedural logic required for programming could in principle apply to many other domains does not tell us much about how it might transfer to those domains. Yet it is knowledge of the "how" of transfer that may allow predicting when transfer will occur and suggest ways to promote transfer. Would more practice with programming always facilitate greater or farther transfer? Is the evocation of metacognitions always necessary? Can playful programming, as when children "mess around" with Logo, be expected to foster transfer? Under what conditions?

This paper addresses the "how" of transfer. Specifically, we present the outlines of a theory of the mechanisms of transfer; we use it to examine the contrasts between certain studies that did and did not obtain positive transfer results from programming; and we discuss what kinds of transfer from programming can be expected and when.

Two Roads to Transfer

What can be said about the mechanisms of transfer by which programming might enhance cognitive functioning generally? Pea and Kurland (1984a) offer some general insight, concluding among other things that (a) different levels of programming proficiency may enable the transfer of different concepts and skills; transfer of programming skills to the solution of nonprogramming problems or to planning might follow only from reasonably advanced levels of programming proficiency that few children actually reach; (b) transfer to problem solving and planning may require considerable metacognition;

consequently, young children might not manifest such transfer; (c) transfer does not occur spontaneously; it requires guidance and modeling.

We fully concur with these conclusions. In themselves they offer a broad interpretation of why transfer often fails to appear. They gain all the more weight if put in the context of an encompassing view of transfer that maps two distinct cognitive routes to transfer, one of which is highlighted by Pea and Kurland's remarks. Prior thinking about transfer has by in large treated transfer as a unitary phenomenon dependent on the spontaneous "spill over" of learning from one context to others according to an ill-defined metric of situational similarity. Elsewhere, we have questioned this view, elaborating an account of transfer that identifies two distinct transfer mechanisms and the somewhat different transfer results associated with them (Perkins & Salomon, in press; Perkins & Salomon, in press). Here we summarize.

"Low road transfer" gives a name to one of the mechanisms. Transfer occurs by the low road when a performance practiced to near automaticity in one context becomes activated spontaneously by stimulus conditions in another context. For example, if you know how to drive a car and sit behind the wheel of a truck, the familiar configuration of steering wheel, windshield, and so on will engage your car driving habits without any mindful effort on your part. Fortunately, those habits suit the occasion reasonably well, so you can drive the truck adequately albeit with some care. Low road transfer often can be negative rather than positive; for another example from driving, in correcting a skid you must steer in the direction of the skid deliberately, contrary to the spontaneous low road transfer from normal driving that presses you powerfully to correct the course of the car directly.

Far-ranging low road transfer occurs only when substantial practice yields automaticity (Schneider & Shiffrin, 1977) and when there is <u>varied</u> practice that samples widely the circumstances that are targets for transfer. Otherwise, the different stimulus characteristics of different contexts work against low road transfer. Consequently, low road transfer to markedly different contexts characteristically happens with behaviors that infuse a range of circumstances, as with literacy or certain social conducts. For example, Luria (1976) found a broad impact of schooling and literacy among the people of Uzbekistan and Kirghizia in the 1930's. In contrast, Scribner and Cole (1981) found little impact of Vai and Arabic literacy on the thinking of those possessing these literacies in the Vai, an African tribe. The conflicting findings are explained when one recognizes that literacy and schooling in general were infiltrating the cultures Luria investigated, whereas Vai and Arabic literacy play very circumscribed roles in the Vai culture according to Scribner and Cole (1981).

High road transfer, in contrast to low road, involves deliberate mindful abstraction from one context and application to another. For example, one might learn the principle, "seek control of the center," from chess play and subsequently generalize and apply it to a business or military context. High road transfer entails re-representing knowledge in a symbol system that affords ready abstraction, such as language, at a greater level of generality that subsumes a greater range of cases. As the chess example intimates, the high road is natural for explicit strategies and principles, such as general strategies of problem solving, decision-making, communication, or learning. Mindfulness in the sense of conscious, deliberate and critical cognitive processing is a key ingredient (Langer, 1989). Self-conscious efforts to

transfer fostered by self-monitoring and recognition of the need to recruit past experience to solve current problems are important (e.g. Belmont, Butterfield, & Ferretti, 1982; Brown, Bransford, Ferrara, & Campione, 1983). Genuine understanding of the abstraction versus rote memory for an abstraction learned in class is crucial, as studies in discovery learning indicate (Haslerud & Meyers, 1958; Kersh, 1958, 1962; McDonald, 1964). Whereas low road transfer is limited by the triggering stimuli that will activate an automatized performance and hence requires varied practice to reach far, high road transfer is limited by the cognitive effort and mindful decontextualization needed and thus requires motivation and some degree of cognitive skill to reach far.

Before addressing potentials of transfer from programming directly, we should note that schooling in general does little to encourage far transfer either by way of the low road or the high road. Students rarely are provoked to think imaginatively about cross-connections between subject matters, exploring generalizations from one subject matter that might apply to another. The high road thus gets travelled rarely. What transfer does occur seems likely to happen by way of the low road. Although, as already noted, a broad impact of literacy and schooling on cognitive performance occurs (Luria, 1976; Scribner & Cole, 1973), in the details of particular subject matters one typically encounters a narrow range of practice that misses opportunities for low road transfer. For example, mathematical problem solving in late elementary school tends to focus on a few traditional classes of problems, such as time - rate - distance, age, and mixture problems. Furthermore, plainly many students do not master a number of important academic performances to the point where they become automatic and hence candidates for low road transfer. In sum, conventional schooling receives a mediocre grade for fostering both low and high road transfer. The question remains whether programming specifically and the ways in which programming is taught and experienced promise and actually yield better results.

Potentials of Transfer from Programming

Before addressing the research literature on transfer from programming, it only makes sense to consider what transfer effects <u>might</u> occur in consequence of programming experience. Such an exploration will put in context those transfer effects that have been investigated empirically and their respective findings. In particular, we can ask whether to this point research on programming and transfer from it has cast a wide net for possible transfer effects or only sampled a few possibilities, one issue to be considered at the end of this paper.

In general, programming is a remarkably rich cognitive enterprise that might yield many different sorts of transfer effects. In order to convey a sense of the range, we have identified six broad categories of transfer that might occur; perhaps there are others as well. After describing the six, we will comment on where high road versus low road transfer is likely to play the greater role and finally on the hazards of forecasting transfer, hazards that make the "might" in the question "what transfer effects might occur" very seriously meant.

<u>Category 1: Mathematical and geometric concepts and principles.</u> Programming with Logo or other languages that readily afford graphic manipulation may build mastery of geometric concepts and associated

mathematical concepts (Papert, 1980). Numerically oriented programming might enhance understanding of the concept of a variable, algebraic expressions, the integer-real number distinction, power notation, and other conventional aspects of mathematics.

Category II: Problem solving, problem finding, and problem management strategies. Included here are such traditional problem solving strategies as attempting to break a problem into parts or relating it to a previously solved problem (Polya, 1954; Polya, 1957; Wickelgren, 1974); representing, defining, and reconsidering the representation and definition of problems (Getzels & Csikszentmihalyi, 1976; Greeno, 1983; Hayes, 1981); managing the solution process for a complex problem (Schoenfeld, 1980). Also included are strategies of diagnostic thinking for debugging, in some ways analogous to, for instance, medical diagnosis (Elstein, Shulman, & Sprafka; Shulman, Loupe, & Piper, 1968), and those for planning, an aspect of programming investigated by Pea (1982).

Category III: Abilities of formal reasoning and representation. The importance of conditional statements in many programs and the necessity to exhaust all possible cases in a properly written program might foster the skills and understandings involved in formal logical tasks such as syllogistic reasoning or using the propositional calculus (Falmagne, 1975; Johnson-Laird, 1983; Mason & Johnson-Laird, 1972). Papert (1980) forecast an impact on combinatorial thinking in the Piagetian sense, where for instance a youngster versed in programming might construct all possible combinations much earlier than usual. In addition, programming provides experience with constructing formal representations of situations, which might transfer to logical and mathematical modeling of situations other than by computer.

Category IV: Models of knowledge, thinking, and learning. Students approach a learning task with tacit theories of knowledge, thinking, and learning that influence their performance, for instance some students taking an "either you get it or you don't" view of learning while others see learning as an incremental process resulting from effort and concentration (Dweck & Bempechat, 1980; Dweck & Licht, 1980). Such traits appear in students' reactions to programming, affecting the way they go about it and what they learn from it (Zelman, 1985, April). Moreover, facilitative interactions with programming, a complex activity that can be mastered to many different degrees and for different sorts of tasks, may change such traits for the better. More directly, programming provides a model for thinking about one's own mind and how one approaches tasks. The notions of specifying a procedure for oneself and of debugging it figure prominently here (Papert, 1980). Having such a model might foster metacognitive awareness and control beyond the context of programming (Clements & Gullo, 1984).

Category V: Cognitive styles. Programming appears to put a high premium on certain cognitive styles, for instance precision (Baron, 1985), reflectivity over impulsivity (Kagan, 1965; Kagan & Kogan, 1970), and field independence over field dependence (Witkin, 1976). Extensive programming experience might "train up" these cognitive styles with resultant spill-over to nonprogramming activities (e.g. Olson, 1985).

Category VI: Enthusiasms and tolerances. To this point, possible transfer effects have been formulated cognitively. But it is also important to recognize that there may be important affective consequences as well. Exposure to a reasonably constructive task, all too rare in schooling, may

kindle some students' enthusiasm for meaningful academic engagement, after which they might find similar opportunities in science projects or writing activities. Complex and engaging as it is, programming might involve some students in prolonged work at high cognitive load levels and consequently lead to habituation to the aversive feel of high cognitive load.

Some general comments about these categories are due. First of all, note the enormous range of potential transfer from programming. Second, it should be added that in our view any rich constructive activity involves a somewhat similar although not identical range. Properly pursued writing activity, for instance, may have many of the same potentials (Bereiter & Scardamalia, 1982), as indeed do design activities of any sort (Perkins, 1984, in press). Third, however, programming provides some opportunities for transfer not offered by many other highly constructive activities, for instance the notion of learning a skill as planning and debugging a logical procedure.

Fourth, the categories have been ordered along a continuum ranging from cognitions volitionally applied, at least by nonexperts (for instance, principles of mathematical problem solving, planning strategies, formal representations of problems) to those usually less under the individual's volitional control (models of knowledge, reflectivity, tolerance) (Kuhl, in press). This rough layout of course allows that skills volitional in the novice may become automatized in the expert and that, with effort, characteristically nonvolitional aspects of mind such as cognitive styles may become subject to conscious, deliberate application. Finally, let us acknowledge that despite our efforts to separate into six categories potential transfers from programming, inevitably there is some overlap. For instance, affective factors alluded to in the last category connect with several of the other categories.

In which categories might high road transfer figure primarily and in which low road transfer? Any answer must be conjectural, not only for lack of data but because each category itself involves a mix of somewhat different elements and, of course, because the same performance can often transfer by way of both the low road and the high road if conditions for both are met. Those caveats mentioned, it seems likely that enthusiasms and tolerances would transfer primarily by way of the low road, if they transfer at all. These are relatively automatic aspects of human behavior in response to appropriate stimulus conditions. Cognitive styles seem candidates both for low and high road transfer. Regarding the low road, stimulus conditions demanding precision in mathematics or other disciplines might trigger habits of, for instance, precision acquired in programming. Regarding the high road, students of programming might to some extent learn to "take themselves in hand," adopting precision, attempting to concentrate, and so on, in addressing a programming task. Such acts of taking oneself in hand might transfer by the high road to other contexts. By in large, performances in the other -- more volitional than habitual -- categories seem to lend themselves especially to high road transfer, as explicit knowledge and intentional strategies rather than stimulus-controlled habits are emphasized.

Finally, how firm a forecast of transfer from programming do the categories provide? Not very firm at all, for two reasons. First of all, the conditions for either high or low road transfer must be met, and, as emphasized in our outline of the high and low road theory, typical schooling does not meet either of them. Whether typical instruction in programming does better will be addressed shortly. But second, even given the conditions

for transfer, predicting what might transfer to what is an uncertain matter. For example, programming requires breaking problems down into subproblems; so does the solving of many mathematical problems or everyday problems. But do they require breaking problems down in the same way? Certainly not transparently. To be sure, the general problem-solving move of trying to break a problem down might transfer, but how empowering is that in itself? Might one even get negative transfer from trying to break problems down in the wrong way, imported inappropriately from programming (Seidman, 1981)?

To generalize the dilemma, merely because the same skill or ability label applies to two tasks one cannot predict transfer with confidence. While two tasks may both involve a precise cognitive style, breaking problems down, knowledge of geometry, high cognitive load, skills of deductive thinking, or whatever, they may engage these in crucially different ways not captured by such holistic labeling. Only a finer grained analysis, preferably based on commitment to cognitive theories of the specific task domains, would say whether the designated skill or ability boils down to the same thing in the two domains sufficiently to predict transfer. While there are partial cognitive theories of certain task domains that might be helpful, for instance of programming and theorem proving (Anderson & Reiser, 1985), mathematical problem solving (Schoenfeld, 1982; Schoenfeld & Herrmann, 1982), and problem solving in physics (Chi, Feltovich, & Glaser, 1981; Larkin, McDermott, Simon, & Simon, 1980), these theories are somewhat provisional themselves; making strong predictions about transfer based on a close technical application of two such theories for the two domains in question seems premature. For the time being, we suggest that it is better to reason broadly about what transfers might occur and examine some prospects empirically to see what does occur. A number of investigators have done just this, although without any general theory of the mechanisms of transfer. We now turn to what some of them have discovered.

The Two Roads in Action: Examination of Research

What roads to transfer have been attempted in research with programming and what were the transfer results? As already indicated, a number of efforts to measure transfer from programming have yielded a confusing mix of occasional positive findings and "no significant difference" (Blume, 1984; Land & Turner, 1985; Pea Kurland, 1984a). Can the present theory distinguish between the successes and the failures? Let us take a close look at a few investigations -- two in which no transfer was observed, one with partial success, and two with marked positive results.

Pea and Kurland: Logo and planning

Pea and Kurland (1984b), in two well-publicized studies, set out to test the hypothesis that children's engagement in Logo programming would have an impact on their planning ability. Planning, argued Pea and Kurland, is very much an integral part of programming -- "that set of activities involved in developing a reusable product consisting of a series of written instructions to make a computer accomplish some task" (p. 8). They reasoned that a well-designed planning task could tap changes in planning ability, whether brought about by planning-in-action during programming or by preplanning.

Thirty-two upper middle class 9 and 12 year olds were either given a twice per week 30 hour exposure to Logo, essentially self-initiated and

self-guided, or belonged to a no-treatment control group. All children were pre- and posttested on an elaborate classroom chore scheduling planning task. Surprisingly, although the Logo students tended to receive somewhat higher scores for planning efficiency, the difference did not reach significance. Nor did the Logo group differ from the control group on measures of plan quality, flexibility, or any other aspect of planning. It may be that 30 hours of child-initiated Logo programming has no transferable effect on high level thinking.

In a second study, Pea and Kurland employed a new planning task, designed to resemble the deep structural features of programming. The study resembled the preceding one except that the teachers took a more directive role in guiding the children's explorations of Logo. Half of the 32 children received Logo instruction for about half a school year; the other half again served as a no-treatment control group. Also, half the children received a planning task in which immediate feedback was provided, encouraging them to notice the similarity to programming.

The hypotheses under test were that the Logo group would show a greater gain in planning ability following their Logo experience, make more and better use of the feedback during the planning task, and take more time for thinking about alternative plans than the control group. In fact, the Logo group showed no better nor more thoughtful planning behavior than the control group on any of the planning measures.

Considered from the standpoint of our perspective on transfer, these findings should come as no surprise. According to Papert (1980), Logo provides a whole micro-environment in which "powerful ideas" are acquired more or less incidentally while self-guided explorations take place, much as a language indigenous to a country is acquired. Such circumstances are a recipe for transfer by way of the low road of extensive and varied practice: Children will become conversant with computational ways of thinking much as they become conversant with their mother tongue, this affecting their thinking "even when they are far removed from . . . a computer" (Papert, 1980, p. 4). But the amount of practice provided in the Pea and Kurland studies, as in most school learning, falls far short of what low road transfer would require. Practice needs to be sufficiently extensive to yield near automaticity. On the contrary, Pea and Kurland themselves document that the students in their experiments achieved only a very limited mastery of the fundamentals of Logo programming itself.

With the low and high roads in mind, the low road setting often employed for Logo may be a mistake even if the students achieved considerable mastery of Logo. The categories of potential transfer presented earlier suggested that the application of planning skills typically would involve volition; their transfer would occur by way of the high road of mindful abstraction and decontextualization. Our general reading of Papert suggests that he envisions a process of mindful transfer. Pea and Kurland (1984b) urged that for programming to affect planning skills the activity "must be supported by teachers who, tacitly or explicitly, know how to foster the development of planning skills through a judicious use of examples, student projects, and direct instruction" (p. 44), which certainly has a high road slant. But nothing that would promote such high road transfer was provided in the studies just described. Although certainly bright students sometimes do some high road transferring by themselves, it is hardly surprising that a setting characteristic of low road transfer does little to provoke use of the high road.

Linn: Problem solving skills

Linn (1985) provides some evidence suggesting that high road transfer from programming can occur if seriously pursued. Linn addressed the effects of learning to program in Basic on problem solving abilities, another set of skills inviting volitional application and falling in the same category in our earlier scheme as planning skills. Specifically, Linn described a chain of three successive achievements in programming that might link problem solving in programming to problem solving in other domains: Mastery of language features, mastery of program design skills, and mastery of programming-related problem solving skills. The higher the programming achievement, Linn implied, the more general the scope of the problem solving skills acquired through programming.

Regrettably, problem solving skills were not assessed independently of programming skill but rather by transfer within that domain. Middle school students receiving at least 12 weeks of programming instruction took a graded test of programming proficiency that included a measure of ability to learn a new programming language. This measure Linn saw as indicative of transfer from programming to problem solving; at the least it would show some transfer beyond the immediate topic of instruction. In all, 600 students participated in the study. Most came from 10 typical classrooms; others came from 3 exemplary ones where teachers had more experience in programming, taught mainly design features of programs, and emphasized either templates (e.g. IF . . . THEN patterns) or procedural skills.

Two findings have particular interest here. First, the exemplary classes displayed programming achievements decidedly higher than did the typical classes; second, achievements in the typical classes correlated highly with general ability whereas those in the exemplary classes did not. A common factor appears to account for both findings: mindful abstraction during the process of learning, either teacher-induced or spontaneously achieved. As to the former, note the description of an exemplary teacher: "She facilitated testing skill by requiring that students attempt to locate "bugs" in their programs for 10 minutes before getting help from an expert," or "she used guided discovery techniques to help students reformulate code when their programs failed to work properly" (Linn, 1985, p. 26). Plainly the exemplary teachers conducted their classes in ways that emphasized mindfulness, thereby facilitating high road transfer.

But high road learning need not always be externally induced. Brighter students are more likely to abstract from the context and discover generalizations that abet transfer than others. This explains the greater correlation between programming achievement and general ability in the typical classes. There, the able students did their own generalizing while the less able ones did not; in the exemplary classes the teachers induced mindful abstraction, reducing the relationship between general ability and achievement.

Of course, this argument amounts to an interpretation of data that might be taken in other ways due to ambiguities inherent in the experimental design. For instance, one would prefer measures of achievement outside the context of programming altogether for any confident conclusion of transfer. The studies to be discussed next provide this.

Clements and Gullo: Impact on cognitive styles and skills

Stronger and clearer evidence for high road transfer from programming appears in two studies, one by Clements and Gullo (1984) and another by Clements (1985). In both studies, young children either received programming instruction in Logo or worked with interactive CAI programs. In the Clements and Gullo (1984) study the Logo instruction lasted about 12 weeks and in Clements (1985) 22 weeks. Assessment of transfer from the programming treatments revealed strong effects on such diverse measures as reflectivity, divergent thinking, and the metacognitive ability of comprehension monitoring (Markman, 1977, 1979), but not on a measure of reading achievement. In one of the studies both first and third graders participated; the findings suggested that the Logo experience had more impact on the younger children (Clements, 1985). In neither study did the CAI treatment show a significant impact on any transfer measure. This serves as a control, showing that mere involvement with a computer cannot explain the gains found in the Logo condition.

Such effects of Logo programming were forecast by the designers of Logo but not found in the controlled experiments conducted by Pea and Kurland. Yet they emerge here. Why? Did Clements' group teach Logo in a different way that could help to realize the alleged potential of Logo for transfer? The answer is a strong affirmative. Logo instruction in studies conducted by Clements and Gullo took place in groups of two to three children guided by an adult tutor, who, it appears, followed Pea and Kurland's (1984b) prescription cited above. Ample evidence argues that the presence of significant others during individual or small group learning, let alone guided instruction, tends to increase learners' mental effort expenditure in processing information; learners become more focussed and mindful, improving both their learning and transfer from it (e.g. Salomon, 1977; Webb & Kenderski, 1984).

Some details of the instruction support this interpretation. In the second study (Clements, 1985), children first planned what they wanted the turtle to draw and then tested each command on the screen. Moreover, when faced with bugs, the children were encouraged to think aloud responses to such questions as "What did you tell the turtle to do? What did it do? What did you want it to do? How could you change your procedure?" (p. 8). While adult participation in this study gradually decreased, work at the computer continued in teams and the children were urged to describe aloud to one another their plans, suggestions, and solutions.

Although we do not know exactly how the children in these studies went about their Logo activities, it is quite clear that the cognitive mechanisms engaged contrasted considerably with the circumstances observed by Pea and Kurland or by Leron (1985). The latter remarked that in the absence of direct instruction most children tended to engage in a "hacking" kind of Logo programming, which is not very conducive to the acquisition of "powerful ideas." Instructional procedures of the style employed by Clements promote focussed mindfulness, apparently provoking high road transfer that yielded the reported transfer results.

Still, one may ask why the CAI conditions in these two studies did not yield similar consequences; adult guidance accompanied the CAI activity as well. Here, the interaction between an activity's potential for cognitive impact and style of presentation requires consideration. As Clements points out, programming encourages the generation of ideas, representation of the

ideas in internal codes that break the ideas down into sequential components, translation of these into a communicable code, the testing of ideas, and their correction. Typical CAI offers hardly anything of the kind. While instruction in the described style could foster mindful abstraction from Logo, CAI provided much less to abstract from.

One further matter merits discussion: Transfer results were neither uniform across tests nor across the two studies. In the Clements and Gullo experiment the difference between the adult-guided experiences of Logo and CAI accounted for 34%-44% of the metacognitive posttest scores (eta square; Rozenthal & Rosnow, 1984), but this difference only explained 18% in the Clements study, which emphasized more open-ended team work. Relatedly, in the Clements and Gullo study strong treatment effects on reflectivity and math achievement appeared, but not at all in the Clements study. On the other hand, about 26% of the variance on the Torrance Test of Creative Thinking was accounted for in the Clements study, compared with only 13% in the Clements and Gullo study. Interestingly, adult-guided Logo experience resulted in higher scores on fluency, originality, and divergence, while team-based experience affected mainly (though more weakly) scores of elaboration. This suggests that not all provisions for high road transfer operate in the same way. Also, greater transfer follows from more focussed mindfulness.

Discussion

Let us stand back from the results reviewed and ponder how to appraise the role of programming in fostering cognitive skills. One might propose four broad stages of development for this idea: the potential of programming is proposed; the potential is proven; the potential is well mapped; the potential is widely realized. On this informal scale, we suggest that the current state of the art places the field somewhere in the second stage: potential proven to a degree but certainly not well mapped nor widely realized.

As reviewed at the outset, a number of individuals for some time have urged that instruction in programming might impact on cognitive skills generally. Until very recently, however, this has remained a proposed potential: Empirical results supporting such transfer have been lacking. Contemporary investigations have carried the field beyond conjecture to demonstrations that such an impact can occur and to some understanding of the conditions that foster it.

In particular, here we have argued that transfer from programming will occur when the conditions of learning allow for either one or both of two mechanisms of transfer. High road transfer mediates a broad cognitive impact of programming by way of deliberate, mindful abstraction of ideas, procedures, skills, and concepts involved in programming, and consequent calculated application in other domains. Low road transfer mediates such an impact by way of varied practice of programming skills to near automaticity, so that other circumstances that make somewhat similar demands spontaneously engage the patterns of cognition in question.

The mixed empirical results concerning transfer from programming accord with this account. Positive transfer results have emerged under conditions affording high road transfer. Some programming experiments have adopted an

instructional style more characteristic of low road transfer but for relatively brief periods of instruction that plainly did not allow for attaining mastery. Under these conditions, no transfer appeared. These findings give grounds for concluding that we have a degree of proof now for the potentials of transfer from programming: Not only are there some positive results but the present framework offers an understanding of why sometimes the results have been negative.

However, for a number of reasons one cannot conclude that the transfer potentials of programming are well mapped. First of all, while the sorts of transfer summarized here range from the specific (math) to more general features of cognitive style (e.g. reflectivity) and illustrate the prospects of transfer, one may question the durability and generalizability of the effects. Lasting changes in reflectivity, metacognitive activity, or creativity after no more than 44 Logo hours? We must recognize the often noted difference between treatment effects under well-controlled experimental versus real-life conditions (e.g., Salomon, 1979). A conservative interpretation of the results to date would say that the studies demonstrate the kinds of transfer that may accrue from programming when provisions for high road learning are made, but not reliably the amounts, distances, or durability of transfer effects. More research will be required to locate such boundaries.

The potentials are not well mapped for another reason also. The six categories of transfer discussed earlier show that the few contemporary experiments by no means explore all the possibilities for transfer. Evidence has been cited of some transfer to mathematics, problem solving, and cognitive styles. Each of these categories is itself complex and in no sense "covered" by the results to date. In addition, no evidence builds an empirical case for transfer to formal reasoning and representation; models of knowledge, thinking, and learning; or enthusiasms and tolerances. Dependent measures simply have not concerned these categories. It should be noted that benefits of programming in all three areas have been forecast and even reported in informal observations (Papert, 1980; Papert, Watt, diSessa, & Weir, 1979; Watt, 1979).

Finally, some comments are due on when and whether the potentials of programming for transfer will be widely realized. We are a long way from this for a variety of reasons. One just reviewed is that the potentials are not well mapped by empirical enquiry yet, but beyond that other concerns emerge. First of all, present results argue that reaping the general cognitive impact of programming requires instruction carefully designed to foster transfer by way of one or another of the two roads. Originally, one might have thought of programming as a kind of cognitive playground: mere engagement in the activity of itself would exercise the mind as real playgrounds exercise young bodies, without any need for instruction finely tuned to provoke such consequences. Unfortunately, the research argues strongly against such a vision. Instead, certain inconvenient conditions must be met.

The conditions for transfer by the low road alone seem too inconvenient. Varied practice to near automaticity appears to require a time commitment to programming instruction that would be out of the question in most educational settings. What also requires instruction that achieves some reasonable degree of competence in programming: One has to have a skill to automatize it. At present, many students drop by the wayside during their first semester of programming. In general, the studies of literacy alluded to earlier suggest

that widespread impact from low road transfer requires the infiltration of numerous aspects of a culture with the skills in question. While it seems unlikely that programming per se will ever become anywhere near as widely learned as literacy, conceivably a number of general ideas and ways of thinking related to information technologies will pervade much of society and come to shape and empower thinking somewhat (Cf. Perkins, 1985). But this is not likely to happen quickly.

In the short term, the high road offers a more likely route but not an easy one. Transfer by the high road benefits from a high teacher-student ratio, Socratic interaction with the learners, great sensitivity on the part of the teacher for the ebb and flow of enthusiasm and understanding in the individual student, calculated provocation of abstraction and connection-making, and so on. Such mediating tactics place substantial demands on the resources of school systems and the skills of teachers, many of whom are new to programming themselves. In the long run, carefully designed curricula, improved programs of teacher training, and similar means may make possible the routine implementation of programming instruction that has an impact on cognitive skills. At present, every such effort appears to be a separate saga full of false starts, unexpected problems, and, most often, unsatisfactory results.

Wide realization in practice of the potentials of transfer from programming remains a vexed question for one more reason as well: Numerous other approaches to fostering the development of cognitive skills rival programming. Many of the individuals caught up in enthusiasm for the mind-expanding effects of programming seem not to be aware that the last decade has seen intensive research and development efforts and a number of teaching experiments, some of them successful, in the general area of fostering cognitive skills. Reviews appear in Chipman, Segal, and Glaser (1985), Nickerson, Perkins, and Smith (1985), and Segal, Chipman, and Glaser (1985), for instance. While current research argues that programming is a way of developing cognitive skills, it is by no means clear which is the best way. Indeed, one might doubt that there is a best way general across individuals, settings, and cognitive objectives.

In summary, recent findings have justified the conjectures of a number of thinkers that programming offers a context for the development of cognitive skills. Moreover, one need not feel disoriented by a pattern of conflicting results. The high road - low road perspective outlined here offers a broad characterization of the conditions under which such results should appear. All this certainly encourages further efforts to study just what gains are possible, how to ensure that gains persist, and how to engineer practical instruction that achieves those gains. At the same time, one does well to realize that these additional problems are by no means trivial -- indeed, they are probably more difficult than testing for an impact of programming in small-scale experiments -- and that programming as a way of developing cognitive skills will have to compete in the intellectual and economic markets with a number of other approaches to the same general problem.

References

Anderson, J. R., & Reiser, B. J. (1985). The LISP tutor. Byte, 10(4), 159-175.

Baron, J. (1985). What kinds of intelligence components are fundamental? In S. S. Chipman, J. W. Segal, & R. Glaser (Eds.), Thinking and learning skills, Volume 2: Current research and open questions. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Belmont, J. M., Butterfield, E. C., & Ferretti, R. P. (1982). To secure transfer of training instruct self-management skills. In D. K. Detterman & R. J. Sternberg (Eds.), How and how much can intelligence be increased? (pp. 147-154). Norwood, New Jersey: Ablex.

Bereiter, C., & Scardamalia, M. (1982). From conversation to composition: The role of instruction in developmental process. In R. Glaser (Ed.), Advances in instructional psychology (pp. 1-64). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Blume, G. W. (1984, April). A review of research on the effects of computer programming on mathematical problem solving. Paper presented at the annual meeting of the American Educational Research Association, New Orleans.

Brown, A. L., Bransford, J. D., Ferrara, R. A., & Campione, J. C. (1983). Learning, remembering and understanding. In J. H. Flavell & E. Markman (Eds.), Cognitive development, 3, of P. H. Mussen (Ed.), Handbook of child psychology (4th ed.). New York: Wiley.

Chi, M., Feltovich, P., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. Cognitive Science, 5, 121-152.

Chipman, S. F., Segal, J. W., & Glaser, R. (Eds.). (1985). Thinking and learning skills, Volume 2: Research and open questions. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Clements, D. H. (1985, April). Effects of Logo programming on cognition, metacognitive skills, and achievement. Presentation at the American Educational Research Association conference, Chicago, Illinois.

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. Journal of Educational Psychology, 76(6), 1051-1058.

Dweck, C. S., & Bempechat, J. (1980). Children's theories of intelligence: Consequences for learning. In S. G. Paris, G. M. Olson, & H. W. Stevenson (Eds.), Learning and motivation in the classroom (pp. 239-256). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Dweck, C. S., & Licht, B. G. (1980). Learned helplessness and intellectual achievement. In J. Garber & M. Seligman (Eds.), Human helplessness. New York: Academic Press.

Elstein, A. S., Shulman, L. S., & Sprafka, S. A. (1978). Medical problem solving: An analysis of clinical reasoning. Cambridge, Massachusetts: Harvard University Press.

Falmagne, R. J. (Ed.). (1975). Reasoning: Representation and process in children and adults. Hillsdale, New Jersey: Erlbaum.

Feurzeig, W., Horwitz, P., & Nickerson, R. (1981). Microcomputers in education (Report no. 4798). Cambridge, Massachusetts: Bolt, Beranek, & Newman.

Getzels, J., & Csikszentmihalyi, M. (1976). The creative vision: A longitudinal study of problem finding in art. New York: John Wiley & Sons.

Greeno, J. G. (1983). Conceptual entities. In D. Gentner & A. L. Stevens (Eds.), Mental models. Hillsdale, New Jersey: Erlbaum.

Haslerud, G. M., & Meyers, S. (1958). The transfer value of given and individually derived principles. Journal of Educational Psychology, 49, 293-298.

Hayes, J. R. (1981). The complete problem solver. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Johnson-Laird, P. N. (1983). Mental models. Cambridge, Massachusetts: Harvard University Press.

Kagan, J. (1965). Individual differences in the resolution of response uncertainty. Journal of Personality and Social Psychology, 2, 154-160.

Kagan, J., & Kogan, N. (1970). Individuality and cognitive performance. In P. Mussen (Ed.), Carmichael's manual of child psychology (Vol. 1). New York: Wiley.

Kersh, B. Y. (1958). The adequacy of 'meaning' as an explanation for the superiority of learning by independent discovery. Journal of Educational

Psychology, 49, 282-292.

Kersh, B. Y. (1962). The motivating effect of learning by directed discovery. Journal of Educational Psychology, 53, 65-71.

Kuhl, J. (in press). Volitional mediators of cognition behavior consistency: Self-regulatory processes and action versus state orientation. In J. Kuhl & J. Beckmann (Eds.), Action control from cognition to behavior. New York: Springer-Verlag.

Land, M. L., & Turner, S. V. (1985). What are the effects of computer programing on cognitive skills? Paper presented at the annual meeting of the Association for Educational Data Systems, Toronto, Canada.

Langer, E. (1980). Rethinking the role of thought in social interaction. In J. Harvey, W. Ickes, & R. Kidd (Eds.), New directions in attribution research (vol. 2). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Modes of competence in solving physics problems. Cognitive Science, 4, 317-345.

Leron, U. (1985). Logo today: Reason and reality. The Computing Teacher, February, 26-32.

Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. Educational Researcher, 14, 14-29.

Luria, A. R. (1976). Cognitive development: Its cultural and social foundations. Cambridge, Massachusetts: Harvard University Press.

Markman, E. M. (1977). Realizing that you don't understand: A preliminary investigation. Child Development, 48, 986-992.

Markman, E. M. (1979). Realizing that you don't understand: Elementary school children's awareness of inconsistencies. Child Development, 50, 643-655.

McDonald, F. J. (1964). Meaningful learning and retention: Task and method variables. Review of Educational Research, 34, 530-544.

Nickerson, R., Perkins, D. N., & Smith, E. (1985). The teaching of thinking. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Olson, D. R. (1986). Computers as tools of the intellect. Educational [illegible]

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York: Basic Books.

Papert, S., Watt, D., diSessa, A., & Weir, S. (1979). Final report of the Brookline LOGO project, part II: Project summary and data analysis (A.I. Memo No. 545, LOGO Memo No. 53). Cambridge, Massachusetts: Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Pea, R. D. (1982). What is planning development the development of? In D. Forbes & M. Greenberg (Eds.), New directions in child development: Children's planning strategies. San Francisco: Jossey-Bass.

Pea, R. D., & Kurland, D. M. (1984a). On the cognitive effects of learning computer programming. New Ideas in Psychology, 2(2), 137-168.

Pea, R. D., & Kurland, D. M. (1984b). Logo programming and the development of planning skills (Report no. 16). New York: Bank Street College.

Perkins, D. N. (1984). Creativity by design. Educational Leadership, 42(1), 18-25.

Perkins, D. N. (1985). The fingertip effect: How information-processing technology changes thinking. Educational Researcher, 14(7), 11-17.

Perkins, D. N. (in press). Knowledge as design. Hillsdale, New Jersey: Lawrence Erlbaum Associates.Witkin, H. A. (1976). Cognitive style in academic performance and in teacher-student relations. In S. Messick & Associates (Eds.), Individuality in learning. San Francisco: Jossey-Bass.

Perkins, D., & Salomon, G. (in press). Transfer and teaching thinking. In Bishop, J., Lochhead, J., & Perkins, D. (Eds.). Thinking: Progress in research and teaching. Hillsdale, New Jersey: Erlbaum.

Polya, G. (1954). Mathematics and plausible reasoning (2 vols.). Princeton, New Jersey: Princeton University Press.

Polya, G. (1957). How to solve it: A new aspect of mathematical method (2nd ed.). Garden City, New York: Doubleday.

Rosenthal, R., & Rosnow, R. L. (1984). Essentials of behavioral research: Methods and data analysis. New York: McGraw Hill.

Salomon, G. (1977). Effects of encouraging Israeli mothers to co-observe "Sesame Street" with their five year olds. Child Development, 48, 1146-1151.

Salomon, G. (1979). Interaction of media, cognition, and learning. San Francisco: Jossey-Bass.

Salomon, G., & Perkins, D. N. (1984, August). Rocky roads to transfer: Rethinking mechanisms of a neglected phenomenon. Paper presented at the Conference on Thinking, Harvard Graduate School of Education, Cambridge, Massachusetts.

Schneider, W., & Shiffrin, R. M. (1977). Controlled and automatic human information processing: I. Detection, search, and attention. Psychological Review, 84, 1-66.

Schoenfeld, A. H. (1980). Teaching problem-solving skills. American Mathematical Monthly, 87, 794-805.

Schoenfeld, A. H. (1982). Measures of problem-solving performance and of problem-solving instruction. Journal for Research in Mathematics Education, 13(1), 31-49.

Schoenfeld, A. H. & Herrmann, D. J. (1982). Problem perception and knowledge structure in expert and novice mathematical problem solvers. Journal of Experimental Psychology: Learning, Memory, and Cognition, 8, 484-494.

Scribner, S. & Cole, M. (1981). The psychology of literacy. Cambridge, Massachusetts: Harvard University Press.

Scribner, S., & Cole, M. (1973). Cognitive consequences of formal and informal education. Science, 182, 553-559.

Segal, J. W., Chipman, S. F., & Glaser, R. (Eds.). (1985). Thinking and learning skills, Volume 1: Relating instruction to research. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Seidman, R. H. (1981, April). The effects of learning a computer programming language on the logical reasoning of school children. Paper presented at the American Educational Research Association conference, New York.

Shulman, L. S., Loupe, M. J., & Piper, R. M. (1968). Studies of the inquiry process (Final report no. 5-0597). Michigan State University.

Wason, P. C., & Johnson-Laird, P. N. (1972). Psychology of reasoning: Structure and content. Cambridge, Massachusetts: Harvard University Press.

Watt, D. (1979). Final report of the Brookline Logo Project. Part III: Profiles of individual students' work (Logo memo no. 54, A.I. memo no. 546). Cambridge, Massachusetts: Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Webb, N. M., & Kenderski, C. (1984). Student interaction and learning in small group whole class settings. In P. L. Peterson, L. C. Wilkinson, & M. Hallinan (Eds.), The social context of instruction: Group organization and group processes. New York: Academic Press.

Wickelgren, W. A. (1974). How to solve problems: Elements of a theory of problems and problem solving. San Francisco: W. H. Freeman and Co.

Witkin, H. A. (1976). Cognitive style in academic performance and in teacher-student relations. In S. Messick & Associates (Eds.), Individuality in learning. San Francisco: Jossey-Bass.

Zoltan, S. (1985, April). Individual differences and the computer learning environment: Motivational constraints to learning LOGO. Presented at the American Educational Research Association Annual Meeting, Chicago, Illinois.