INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

COMPUTER PROGRAMMING AND ANALOGICAL REASONING: AN EXPLORATORY STUDY

by

MARCIA J. SCHLAFMITZ

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1996

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Nov 15 '95
Date

T C Wendbergen
Chair of the Examining Committee

Nov 15 '95
Date

Executive Officer

Frank Beckman

Bonnie Brownstein

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Dedicated to the Memory of My Father

Alex Schlafmitz

who taught me to value education
above all

Acknowledgements

My warm thanks go to my dissertation adviser, Professor Thomas C. Wesselkamper, former Executive Officer of the computer science program at the Graduate School of the City University of New York and Chairman of the Hunter College department of computer science. Tom's support, encouragement, and willingness to discuss matters of both content and process were essential to my undertaking and completing this dissertation. My thanks also go to Professor Stanley Habib, current Executive Officer of the graduate program, for his support and encouragement. Without the inspiration of the late Professor Charles T. Weldon of City College I might never have learned what a profoundly exciting discipline computer science can be. Thank you Charlie.

My deep appreciation goes to my mother for her ever-present love and for the values she taught me; to Arnold Lieber and Jay Harris for always believing in me; and to Steve Whetstone for his patience, humor, and love.

Table of Contents

I. Project Overview

This project addresses several questions about the intellectual skills involved in computer programming. It is motivated in part by discussions in the report on Integrity in the College Curriculum of the Association of American Colleges (1990). The Integrity report resulted from a study, begun in 1982, of what was sensed to be the "decline and devaluation" of the undergraduate liberal arts degree. Among the report's recommendations for reversing this trend in undergraduate education was the implementation of what it refers to as "a new area of research ... indigenous to specific subject areas,"[1] whose goal is the analysis of the specifics of the learning process in a given academic discipline. The report views this research as most effective when it is carried out by investigators whose primary expertise is in the given subject area.

This new type of research is presented as integral to the development of a much needed curriculum containing subject matter courses that go beyond a concern only for "coverage and factual knowledge" to an exploration of "methods and processes ... [of] inquiry, abstract logical thinking, [and] critical analysis [that] are at the heart of the intellectual process." A desire to examine the possibility of integrating the teaching of general methods of thought into the computer science curriculum thus informs

---

[1]Emphasis added.

the design of this study.

Additional motivation for this project arises from the emergence in the last decade of the 20th century of issues that, while present earlier, could remain in the background when the discipline of computer science was less mature and the computerization of the workplace less advanced. Typical is Labor Secretary Robert Reich's analysis of the workplace of the 21st century (Reich 1991). Reich clearly identifies the category of worker who will not only be in worldwide demand and well paid, but will also have some control over working conditions. He calls these workers "symbolic analysts." A symbolic analyst earns a living solving and identifying problems and brokering solutions by manipulating symbols. Symbolic analysts "simplify reality into abstract images that can be rearranged, juggled, experimented with, communicated to other specialists, and then, eventually, transformed back into reality." As Reich describes what he considers the proper education for a future symbolic analyst, he states his concern that "no more than 15 to 20 percent" of American students are being adequately prepared for "a lifetime of symbolic-analytic work."

The effective and creative use of the computer, the ultimate analyzer and manipulator of symbols, is one aspect of education relevant to students' future functioning in this world of symbolic analysis. While economic and social factors must be considered along with educational questions

in any comprehensive discussion of why so few students are acquiring adequate skills, I believe that there are aspects of the problem that may yield to an analysis of the educational methodologies currently in use in the teaching of computer science and other technologically relevant disciplines.

In the spirit of the AAC Integrity report, this project attempts to integrate the teaching of one general method of abstract thinking, analogical reasoning, into an introductory computer-language course. It also looks at Case Western Reserve researcher David Kolb's experiential learning theory (Kolb 1984), which, I believe, may be particularly helpful in the design of educational environments to prepare students to function effectively in a computer-technology-driven society.

To provide a background for these discussions it is necessary to examine several areas of research literature.

II. Literature Review

II.1 Theoretical Background

II.1.1 Introduction

Even a cursory historical survey of the variety of writers who have commented on the forms and uses of analogy reveals its significance in Western thought. Discussions are found from Aristotle through St. Thomas Aquinas to Mill, Russell, and twentieth century philosophers of science (Hesse 1966). Physicist J. Robert Oppenheimer (1956) stated that, from his perspective, "analogy is ... an indispensable and inevitable tool for scientific progress ... [it is] a special kind of similarity ... between two sets of structures, two sets of particulars, that are manifestly very different but have structural parallels."[2]

Popular scientific uses of analogy are familiar. Newton's analogy between the moon and an apple; Descartes's between light beams and tennis balls; Harvey's description of the heart as a pump; Kekulé's model of the carbon ring as a circle of rats; and the Rutherford/Bohr solar-system atom are some of the more well known. Analogy also plays a significant part in two seminal papers on the nature of computation. Turing (1936) forms an analogy between the operations performed by "a man in the process of computing a real number" and the functioning of "a machine which is only capable of a finite number of conditions"; and Post

---

[2]Emphasis added.

(1936) analogizes the human solution of an abstract problem in logic to "a two-way infinite sequence of ... boxes" in which a hypothetical one-dimensional worker marks and erases strokes.

Discussion of analogy in science is one aspect of the more general study of analogy as it is involved in problem solving.[3] As Kantowski (1974) points out in her extensive study of mathematical problem solving, a problem exists for an individual when he or she encounters a situation where the solution is not apparent given the information immediately available. In a precise discipline such as mathematics or computer science, finding a solution to a problem requires finding an algorithm to direct the steps to the desired result. Writing a computer program is a method of problem solving that combines finding an appropriate algorithm and implementing it correctly on a computer. It is therefore research on analogy as it is related to problem solving that most directly informs this project. Literature on analogical reasoning and computer science is discussed first, followed by recent relevant general literature on analogy, and then by that literature that deals with analogical reasoning and problem solving.

---

[3]Even though many discussions of analogy contain discussions of metaphor, the analysis of metaphor is complex in its own right and is not dealt with in this project.

II.1.2 Analogical Reasoning

II.1.2.1 Analogical Reasoning and Computer Science

Analogical processes have been studied by many researchers with interests in computer science. Evans (1968), Winston (1980), Anderson (1983, 1989), Pirolli (1985), and Falkenhainer, Forbus, and Gentner (1989) are some examples. Most of this work has been done in connection with artificial intelligence, in particular in the areas of the computer simulation of cognitive processes, intelligent tutoring systems, and machine learning.

Artificial intelligence researcher John Anderson and his team at Carnegie-Mellon have done extensive work on the theory and design of intelligent tutoring systems for teaching programming languages. Anderson (1989) believes that "people learn problem-solving skills in domains like mathematics and computer programming by analogy." His team's work on the diagnostic-module components of their intelligent tutoring systems assumes that "systematic pattern errors that occur in students' interactions with intelligent tutors ... arise through the [misapplication of] the analogy process." Their detailed production-system models of the analogical-reasoning process as it is applied during the writing and debugging of computer programs are integral to the intelligent tutoring systems they construct. Anderson argues that "the implication is that not only errors but successful learning occurs by analogy."

II.1.2.2 General Literature on Analogy

Recent literature on analogical thinking analyzes the nature of analogy formation as well as salient characteristics of analogies and their use. Gentner, who does research in the cognitive science aspect of artificial intelligence, presents a general theory (structure mapping) of how analogies are constructed (1983). The theory was developed as an attempt to "describe how the meaning of an analogy is derived from the meaning of its parts." In the structure-mapping theory an analogy is seen as "a comparison in which relational predicates, but few or no object attributes, [are] ... mapped from [a] base [domain] to [a] target [domain]." (See Figure 1.) In this theory, the systematicity principle predicts which predicates of the base domain will be mapped to the target domain: "A predicate that belongs to a mappable system of mutually interconnecting relationships is more likely to be imported into the target [domain] than is an isolated predicate." Gentner employs semantic nets in her discussions of structure mapping. VanLehn and Brown (1980) devise what they refer to as planning nets in their very interesting development of a metric for measuring the "closeness" of analogous procedures. The nodes in a planning net are flowcharts; they "are linked only if the application of some constraint or heuristic" to one flowchart results in the other.

Gentner and Gentner (1983) investigate whether "the inferences people make in a topic domain [actually] vary according to the analogies they use." Since "people's understanding of their own mental processes is not always correct ... it could be that ... the underlying thought processes proceed independently of analogy and that analogies merely provide a convenient terminology for the results of the process." Their results support "a true conceptual role for analogical models." (See Section III.1, below.)

Sternberg's componential theory of analogy (1977) is best understood as a "micro" theory. He is interested in identifying the details of the process, which he breaks down into four major components: encoding, inferring, mapping, and applying. Encoding involves identifying the candidates for mapping in both the base and target domains. Inferring refers to identifying the applicable predicates in the base. Mapping finds the elements in the target domain that correspond to the relevant elements in the base. And applying involves the formation of analogous predicates in the target domain that conform to the constraints of the target domain.

II.1.2.3 Analogical Reasoning and Problem Solving

Current research on analogy and problem solving grows out of work done by Gestalt psychologists in the first half of the 20th century (Holyoak 1984). It was incorporated into cognitive science as the discipline emerged in the late

1950s (Gardner 1985), and has been of growing interest to artificial intelligence researchers during the 1970s, '80s, and '90s.

At the University of Michigan in the early 1980s, Gick and Holyoak initiated a line of empirical investigation into analogical problem solving that has been continued and expanded upon by many researchers over the past fifteen years. This work is based on the belief that analogical thinking first develops in early childhood, that it is a significant component of the mental processes involved in reasoning and problem solving throughout life, and that it is a skill that can be improved by training (Holyoak 1984). Gick's and Holyoak's work (1980; 1983) deals with the access, retrieval, and mapping of solutions to analogical problems that are embedded in story contexts. The subjects reading the stories (and therefore challenged to find and apply the analogies) are college students.

Holyoak (1984) writes that since "the function of analogy is to allow transfer of knowledge from a known situation to a novel one," successful analogical problem-solving requires "the ability to identify analogous elements embedded in representations that also include elements that are nonanalogous." He continues, "[Often] knowledge derived by analogy ... must be augmented by additional information in order to achieve a complete solution to the target problem." Gick and Holyoak relate their understanding of

analogical problem solving to the Gestalt psychology tenet that a problem is solved through a reorganization of the elements of the problem in a new way to provide a "structural understanding ... of how all the parts of the problem fit together to satisfy the requirements of the goal" (Mayer 1992).

Gick and Holyoak propose that in analogical problem solving the lexicon of the base-problem domain is restructured, by means of an abstract schema[4] that contains the core ideas of the analog, into the lexicon of the target-problem domain. The restructuring process consists of identifying and mapping both the identities and the structure-preserving differences between the two domains, while, at the same time, factoring out and eliminating those factors specific only to one domain. For example, Polya (1957) suggests that students be prompted to solve the problem of finding the center of gravity of a homogeneous tetrahedron by analogy with the solution to the problem of finding the center of gravity of a homogeneous triangle. The (n=2 dimensional) triangle maps to the (n=3 dimensional) tetrahedron via the structure-preserving predicate "simplest n-dimensional closed straight-lined bounded figure"; the concepts center-of-gravity and homogeneity are mapped as

---

[4]A schema can be thought of as a general knowledge structure used in comprehension (Mayer 1992). Artificial intelligence research makes extensive use of the schema concept; frames and scripts are two AI uses of schemas.

identities; and the differences between plane and solid figures are residual factors specific to each domain separately that are not part of the mappings.

Many researchers have been working to understand the access, retrieval, and mapping of analogies by college students in an attempt to improve college students' problem-solving and inferential capacities. They have often drawn on the original Gick-Holyoak work on analogies in narrative contexts and have used the same or similar narratives (Holyoak 1984; Holyoak and Koh 1987; Ratterman and Gentner 1987; Clement and Gentner 1991; Clement 1994; as well as others). The same analogical-reasoning narratives have been used by some artificial-intelligence researchers who are studying machine learning in an attempt to "emulate analogical processing on computers to produce more flexible reasoning and learning systems" (Falkenhainer, Forbus, and Gentner 1989).

II.1.3 The Cognitive Processes Involved in Programming

II.1.3.1 Overview

Shneiderman and Mayer (1979) postulate what is a generally accepted model for the representation of programming knowledge that draws on the tradition of "information-processing psychology" (Simon 1979). Symbols that can be combined into larger and more complex structures are the basic building-blocks of this model; the symbols reside in structures and are manipulated by "elementary

information processes." According to Simon, "symbol structures and elementary information processes are abstract entities [that are] postulated in order to explain parsimoniously a whole host of observable phenomena." Programming knowledge is considered to be the collection of symbol structures and information processes residing in a memory structure that is composed of short-term memory, long-term memory, and working memory. Short-term memory is a temporary store with a small capacity[5] into which information from the outside world enters; long-term memory is a permanent, large store of already existing knowledge; and working memory is a store in which information from short-term memory is joined with that from long-term memory, integrated, and built upon.

The Shneiderman-Mayer model makes a qualitative differentiation between two types of programming knowledge stored in long-term memory. Semantic programming knowledge refers to a user's mental model of the functioning of a computer. It is general, independent of given programming languages, and ranges from "low-level notions of what an assignment statement does ... to higher-level strategies such as ... recursion by stack manipulation." Syntactic knowledge about a specific programming language, on the

---

[5]A size of about seven mental chunks (recognizable configurations) was proposed for short-term memory in a classic paper on the human brain's information-processing capacity (Miller 1956). Simon suggests that short-term memory can hold about four chunks (1980).

other hand, is "precise, detailed, and arbitrary," involving such things as rules for forming identifiers, iteration-statement formats, and placement of procedure declarations.

According to this model, "semantic knowledge is acquired largely through intellectually demanding, meaningful learning," in the course of which the learner brings the new material into short-term memory and then activates a search of long-term memory for "appropriate ... ideational scaffolding" into which the new material can be assimilated (Mayer 1981). Syntactic knowledge, on the other hand, is "stored by rote and ... not well integrated within existing ... knowledge" (Shneiderman and Mayer 1979). In later work, Mayer (1992) refines the definition of the essential types of programming knowledge to add categories that he refers to as schematic and strategic knowledge. Schematic knowledge refers to the appropriate categorization of routines for different purposes such as looping and sorting; strategic knowledge refers to long-range program planning and debugging techniques. Neither of the latter two categories is dealt with in this project.

Once a learner has some programming knowledge, the model makes use of both the Newell-Simon (1972) given state/goal state concept and Wirth's (1971) stepwise-refinement process to explain the actual construction of a program to solve a proposed problem. The given problem description enters working memory from short-term memory;

generalized knowledge from long-term memory is invoked in working memory to represent the problem in terms of given and goal states. Semantic and syntactic programming knowledge from long-term memory are then mobilized to formulate a representation for the programming solution, an "internal semantics" of the solution. The internal semantics is then seen as evolving, by a stepwise-refinement process, "from a very general [plan], to a more specific plan, to a specific generation of code focusing on minute detail."

II.1.3.2 Why computer programming may be particularly well suited for teaching analogical reasoning

As has been discussed above, "the process of analogical mapping ... can readily be viewed as a mechanism for restructuring one problem in terms of another" Holyoak (1984). The essence of writing a computer program is, similarly, an exercise in restructuring; a problem, represented in its "problem space" must be mapped to the "program space." Consider, for example, the simple problem, often given as the first program in an introductory computer-language class, of writing a program to convert a given number of inches to an equivalent number of centimeters. For a successful program to be constructed, the initial representation of this problem in the "space" of particular measuring units and physical measurements must be restructured through the mediation of an appropriate

computer language into the "space" of computer memory locations, storage and retrieval operations, and fixed- and floating-point register arithmetic. Moreover, as Clement et al. (1986) point out, the semantic knowledge units involved in programming that were identified by Shneiderman and Mayer (operations, algorithms, routines) are "context-independent problem-solving structures," and thus may be "particularly suited to analogical transfer across problems."

II.1.4 Experiential Learning Theory and the Learning Style Inventory

The Learning Style Inventory, which was administered to the students who participated in this project, was developed as part of the experiential learning theory formulated by Case Western Reserve researcher David Kolb. The theory (1984) evolved from Kolb's belief that in the 19th and early 20th century's "overeager embrace of the rational, scientific, and technological, our learning process itself was distorted first by rationalism and later by behaviorism. We lost touch with our own experience as [a] source of personal learning and development." To Kolb's mind this impeding of the learning process is disastrous in "the emerging global village, where events in places we have barely heard of quickly disrupt our daily lives ... [T]he dizzying rate of change and the exponential growth of knowledge all generate nearly overwhelming needs to learn just to survive ... We must [therefore] explore all possible

techniques for improving our capacities for life-long learning." He believes that an understanding of experiential learning theory can aid in the development of teaching methods to help people with varied native approaches to learning survive in a technology-driven rapidly changing environment.

Kolb's theory draws on several philosophical and psychological traditions: Dewey's philosophical writings on education; Piaget's cognitive development theories; and Lewin's work in social psychology. For Kolb, the importance of Dewey's views comes from his emphasis on "learning as a dialectic process integrating experience [with] concepts, observations, and action; the impulse of experience gives ideas their moving force, and ideas give direction to impulse." In Piaget's well-known work on cognitive development, Kolb emphasizes the exposition of the processes of accommodation and assimilation: Learning occurs dialectically as concepts or schemas are accommodated to experience in the world and events and experiences from the world are assimilated into existing concepts and schemas. From Lewin, Kolb takes the centrality of immediate, personal experience as the focal point for learning. For Kolb, this "giv[es] life, texture, and subjective personal meaning to abstract concepts and at the same time provid[es] a concrete, publicly shared reference point for testing the implications and validity of ideas created during the

learning process."

For the purposes of experiential learning theory, Kolb proposes the following working definition: "Learning is the process whereby knowledge is created through the transformation of experience." Most significant for Kolb are the emphasis on process as opposed to outcome and the view of knowledge as arising through active interaction, not as an external entity that is passively absorbed.

II.1.4.1 The Four Styles of Learning

The four styles of learning described by Kolb are complementary, not conflicting. He proposes that: "the learning process is not identical for all human beings ... [because] unique individual adaptive processes [arise] that tend to emphasize some adaptive orientations over others." The theory presents two underlying structural dimensions for the learning process. (See Figure 2.) The prehension dimension describes two dialectically opposed modes of grasping experience: direct apprehension of immediate concrete experience and indirect comprehension of symbolic representations of experience. The transformation dimension describes two dialectically opposed modes of transforming experience: through intentional reflection and through extensional action. The theory holds that while all learners employ varying combinations of these learning styles at different times, one mode tends to predominate.

From these orthogonal dimensions, Kolb identifies four elementary learning styles. Divergent learning grasps knowledge through concrete experience and transforms it through reflective observation; assimilative learning grasps knowledge through abstract conceptualization and transforms it via reflective observation; convergent learning grasps knowledge through abstract conceptualization and transforms it through active experimentation; and accommodative learning grasps knowledge through concrete experience and transforms it via active experimentation.

II.1.4.2 The Learning Style Inventory

The Learning Style Inventory (LSI) is a twelve-item self-description questionnaire designed "to assess individual orientations toward learning" (Appendix H). Kolb explains, "each item asks the respondent to rank-order four words in a way that best describes his or her learning style. One word in each item corresponds to one of the four learning modes ... concrete experience (CE), reflective observations (RO), abstract conceptualization (AC), active experimentation (AE)." To construct a snapshot of a student's predominant style of learning at the time of completion of the LSI, two combination scores are calculated: the difference (AC - CE) measures the extent to which someone emphasizes abstractness over concreteness; the difference (AE - RO) measures the extent to which someone emphasizes action over reflection. These two values

are used as coordinates to plot a data point on a grid with (AC - CE) values as the vertical axis and (AE - RO) values as the horizontal axis. (See Figure 3.) The quadrant in which the data point falls labels the predominant learning style. Crucial to Kolb's theory is the concept that a person's learning style as given by the LSI is fluid. It is meant to be used by educators and students as a guide for understanding and in no way should be thought of as normative. This idea is well formulated in the instruction section of the LSI interpretive booklet (McBer & Co.): "Understanding your learning style helps you become aware of your strengths in some steps of the learning cycle. One way you can improve your learning effectiveness is to use those strengths when you are called upon to learn. More importantly, you can increase your effectiveness as a learner by improving your use of the steps you underuse."

II.2 Pragmatic Background

II.2.1 The Teaching of Thinking Skills and the Transfer of Learning

Dissatisfaction with the ability of many college students to reason clearly and abstractly has led to the production of an extensive literature on the teaching of thinking skills to college students. Much of the literature deals with questions of which skills should be taught and whether or not they can be taught independently of a specific subject area (Nummedal 1987).

Halpern (1987) conducted a national survey that found that virtually every college-level thinking-skills course included instruction on the use of analogies. She suggests that this ubiquity may arise because, as Sternberg says, "Reasoning by analogy is pervasive in everyday experience and would seem to be an important part of what we commonly refer to as intelligence" (1977). Halpern's research yielded positive results on teaching students to use analogies to aid in comprehension, recall, and problem solving.

The issue of the extent to which thinking skills are subject specific is part of a larger debate over the question of the transfer of thinking skills. Glaser (1984), basing his argument on analyses of expert/novice problem solving, concludes that thinking skills should not be taught "as subsequent add-ons." McPeak (1990) argues that while there may be some very limited general thinking skills, "we do not possess an "executive skill" to guide the application of the general skill in a new situation.

Perkins' and Salomon's 1989 survey article "Are Cognitive Skills Context Bound?" presents a historical analysis of the question. They trace it from the "Golden Age of General Heuristics" of the 1950s and '60s[6] with its belief that "local knowledge ... specific to a domain ... was not very important"; through the expert/novice studies,

---

[6]This was the time of the Newell-Shaw-Simon General Problem Solver in early artificial intelligence work.

motivated by the development of artificial intelligence in the late '60s and '70s, that indicated that expert problem solving required a rich, elaborate, specific knowledge base; to what they call the "Skeleton of a Synthesis" of the late 1980s. This synthesis relies on results that show that the question is not either/or; rather "there are general cognitive skills, but they always function in contextualized ways." Moreover, they conclude, that "general heuristics that fail to make contact with a rich domain-specific knowledge base are weak; but when a domain-specific knowledge base operates without general heuristics, it is brittle."

With respect to the transfer of generalized skills, Perkins and Salomon continue, recent results indicate that it "is possible[;] that it is very much a matter of <u>how</u> the knowledge and skill are <u>acquired,</u>[7] and how the individual, now facing a new situation, goes about trying to handle it." Transfer seems to be aided by specific teaching techniques. These include: teaching general heuristics "in a very contextualized way, so the heuristics make good contact with students' knowledge base in the domain"; and encouraging students to generate abstractions and actively decontextualize the generalized heuristic knowledge.

---

[7]Emphasis added.

II.2.2 Teaching Computer Programming and Analogical Reasoning

Two studies that report on the teaching of computer programming and analogical reasoning (Clement et al. 1986; Grandgenett and Thompson 1991) relate directly to the current project. Clement and her colleagues studied the correlation between performance in the computer language Logo and analogical-reasoning tasks. They found that most students in their small sample (N=17) had difficulty with analogical structure mapping from a base problem to a target problem without any prompting, i.e., they had difficulty in noticing an analogy. However, when prompted to the existence of the analogy, ten students carried out a complete structure mapping, and four a partial structure mapping. A Logo proficiency test, one of whose tasks measured the reuse of subprocedures for analogous programming tasks, was administered. They found that, for this sample, "the ability for analogical structure mapping was significantly related to the ability to write reusable modular procedures."

Grandgenett and Thompson (1991) discuss the design of an experimental curriculum for teaching analogical reasoning to novice programmers through a twelve-hour introductory Logo programming course. In this curriculum, the output and corresponding Logo-language computer code for a previously solved programming problem is paired with the desired output

and missing computer code for a new programming problem (Appendix A).

The analogical-reasoning instruction is based on Sternberg's microanalysis of the processes involved. (See Section II.1.2.2, above.) "In the encoding step, students carefully listed all general characteristics of each of the parts of the ... [Output1 : Code1 :: Output2 : Missing Code] programming analogy. In the inferring stage, students traced through the code of the old program to try to review carefully how it produced its corresponding ... output. In the mapping stage, students looked at the similarities and differences between the previous program output and the desired output ... [I]n the applying stage, students attempted to create programming code that produced the desired [new] output ... [T]he reasoning steps were accomplished in a paper- and-pencil format, with students later refining and testing their code on the computer."

Grandgenett and Thompson used as a measure of far transfer (i.e., analogical reasoning development outside programming) the NonVerbal Battery of the Cognitive Ability Test, which uses geometric-analogy problems (Grandgenett 1989). They did not get a statistically significant far-transfer effect for the group as a whole. However, further analysis indicated a positive effect of the guided instruction for freshmen, a negative effect for juniors.

III. Pascal Programming and Analogical Reasoning: An Exploratory Study

III.1 Introduction

This study was designed to see if students in an introductory class in the programming language Pascal could be taught to notice analogies in narrative passages and then apply the analogies in problem solving. Noticing has been isolated as a crucial skill by researchers investigating college students' capacities to make use of analogies for generating solutions to problems (Gick and Holyoak 1980, 1983; Clement 1994; Halpern 1987). Put simply, an analogy can not be used unless it is first noticed on some level. That analogies, once noticed, are useful in problem solving, as speculated by Polya (1957), has been demonstrated empirically in several studies.

In Gentner and Gentner's most interesting 1983 study, there were clear differences in students' proposed solutions to physics problems involving serial and parallel combinations of batteries and resistors depending on the base analogy the students' used for an electric current. As predicted by the researchers, those students whose base analogy for an electric current was a flowing fluid understood the workings of combinations of batteries better than they did combinations of resistors, whereas the reverse was true for those students whose base analogy for an electric current was a moving crowd. Extensive work by Gick, Holyoak, and others with students ranging in age from

preschool through college also provides evidence that the particular base analog presented for a problem influences the solution formulated for the problem (Holyoak 1984; Holyoak et al. 1984).

There are, of course, students who have had rich educational experiences in which the curriculum was infused with abstract intellectual activities. Many of these students may notice and use analogies with facility. However, it is widely agreed (Glaser 1984; Halpern 1987; among many others) that many, if not most, contemporary college students are the products of educational environments that are severely lacking in the type of intellectual activity that can promote the development of abstract-thinking skills. Nummedal (1987) reports that her survey of studies of the reasoning abilities of college students in the 1980s indicates that less than 50% of college students "are able to use formal reasoning processes confidently and reliably." With respect to analogical reasoning, Gick and Holyoak and subsequent researchers following similar protocols have found that "the process of analogical problem solving is neither automatic nor invariably applied by college students" (Gick and Holyoak 1980). It thus seems relevant to ask whether it is a reasonable goal of education to attempt to teach the noticing of apt analogies and their application to problem-solving situations to students to whom it does not come

readily. This project considers that question in the context of an introductory programming-language course.

I refer to this study as exploratory since its goal is the formulation of research questions to be used in the future in a formal, controlled experiment. As a consequence, there is an ad hoc aspect to its methodology that arose both from the nature of the experimental conditions as well as from the fact that, as stated in the Integrity report (1990), the study of learning processes specific to a given discipline is a new form of research for which few paradigms exist. In addition, as Salomon and Perkins (1987) discuss, the theoretical foundations for studies of transfer of skills from one domain to another is in the early stages of formulation. Thus they suggest that at this point of theory development, it makes sense "to reason broadly about what transfers might occur and examine some prospects empirically to see what does occur."

III.2 Subjects

The subjects of this study were students registered in a course called "Introduction to Programming with Pascal" in the Computer Information Systems (CIS) department of Upsala College, a small liberal arts college in East Orange, New Jersey. The course was designed to be the first course in a major sequence in computer information systems.

The formal prerequisites for this course were successful completion of a half-semester course in college

algebra or the equivalent (i.e., a working knowledge of algebra demonstrated on the New Jersey College Basic Skills Placement Test), and completion of a common curriculum hands-on "computer literacy" course, required of all students, which introduces students to word processing, spreadsheets, and databases. In actuality, however, while all students had completed the introductory computing course, the registration system in use at Upsala at the time did not allow for an accurate check on whether students had met the algebra prerequisite. In the course of informal discussions during the semester it became clear to me that some students had not already demonstrated algebra proficiency.

Most students registered for this course with the hope (whether realistic or not) of becoming CIS majors. A few registered because it was required or recommended by another major. Table I, which was compiled from students' responses to the student background questionnaire in Appendix C, handed out at the second class meeting, indicates proposed majors and career goals for the students who responded to those questions.

In the Spring 1994 semester, the level of academic accomplishment of the student body at Upsala College was extremely diverse. While there were students with excellent academic ability and background in the college, they were a definite minority. Many students enrolled in the college

had very weak academic preparation. These characteristics of the college population as a whole were reflected in the introductory programming class.

During the first class meeting, on January 20, 1994, I gave the students a very brief description of the research project in which I was asking them to participate. The only questions the students asked dealt with whether participation would require any time outside of class. Once assured that all of the necessary work would be done during class meetings, all agreed to participate. I then distributed the student participation form (Appendix B), which was completed by all students who were present. The students were told to place their completed forms in an envelope. The last student to complete the form sealed the envelope. This procedure was followed because initially I thought that maintaining student confidentiality would be important in conducting this study. (The student participation form contains both a student's name and code.)

Class absence of weaker students was a college-wide problem at Upsala. Examination of attendance records and of Table I shows that on January 25, 1994, the second class meeting, at which the student background questionnaire (Appendix C) was completed, only 17 of the 22 students registered for the class attended. In addition, during the Spring 1994 semester, the northeastern New Jersey region that includes East Orange experienced an unprecedented 18

snowstorms. The bad weather seriously affected driving conditions for commuting students and had an overall deleterious effect on student morale, further lowering schoolwide class attendance. As a consequence of these factors, throughout the semester, there was intermittent attendance by many students.

Therefore, complete data for this study is available for only eight students. (See Table I, footnote 1.) A student's data is complete if he or she completed the student background questionnaire (Appendix C); the Learning Style Inventory (Appendix H); the pretest learning and analogy sessions (Appendix F); and the posttest learning and analogy sessions (Appendix G). Data for two other students is lacking only one component, the pretest learning session (see Table I, footnote 2). The summary data discussed in Chapter IV therefore concerns only ten students: the eight for whom complete data is available and the two for whom only the pretest learning session is missing. In addition, as the number of students regularly attending class diminished and the informal nature of the study evolved, I became aware of the students' codes. However, I continued using the students' codes to simplify identification and discussion.

III.3 Experimental Design

The overall design of this study involved integration of discussions of analogical reasoning and the concept of

programming by analogy into the introductory Pascal curriculum and the administration of several measuring instruments during the course of the semester.

III.3.1 Incorporating Analogical Reasoning into the Curriculum

Since my goal is an examination of the possibility of incorporating instruction in analogical reasoning into a standard introductory programming-language course, the overall course syllabus was unchanged from previous semesters. A standard introductory Pascal textbook,[8] which had been used during the previous semester, was used again.

III.3.1.1 Introduction of Analogical Reasoning

The concept of analogical reasoning was introduced during the second class meeting. I gave a formal presentation on the concept and some uses of analogy that was based on Grandgenett's model (1989). The presentation included a basic definition of analogical reasoning and examples of the use of analogies. (See Appendix I.) Class discussion followed.

Most of the students had some familiarity with the Rutherford/Bohr model of the atom (and the chemistry major/Dean's List student of course had a sophisticated comprehension of it), some of the students had only a vague notion about it, and a few seemed not to have encountered it

---

[8]Julien Hennefeld, Using Turbo Pascal 4.0-6.0, 2nd. ed., PWS Kent, 1992.

at all before. The majority of the students had never considered the possibility that there might be other models of the solar system, or, in a broader sense, that the sun-centered model was in fact a model, albeit one backed by experimental evidence. Very few of the students knew of solar-system models of any atoms other than hydrogen, nor had they thought about the possible relationship between a sun-centered model of the solar system and a nucleus-centered model of the atom. I tried to focus class discussion specifically on the analogous and disanalogous aspects of the two models.

While none of the students was aware of the historical importance of the human heart/water pump analogy they seemed to comprehend it readily and were eager to point out the important differences, namely, that the heart is clearly not made out of metal. In retrospect, it is clear to me that the possible connections between writing computer programs using an analogical method and other forms of analogical thinking should have been introduced at this time.

III.3.1.2 Subsequent Class Instruction

From the second week of the semester on, class lectures followed the textbook closely. The standard sequence of introductory programming topics were introduced: simple Pascal syntax; very simple sequential programs; input and output statements; the IF-THEN and IF-THEN-ELSE simple decision structures; the CASE multiway decision structure;

the WHILE-DO, FOR-DO, and REPEAT-UNTIL repetition structures; programs involving repetition; modular programs containing non-nested procedures and non-nested functions; Pascal parameter types and rules for parameter passing.

Appendix D gives some of the programming examples that were presented in class. These examples were deliberately taken from the textbook since I believe that instruction for students at this introductory level must be grounded in a textbook that they can consult at any time. The examples were initially introduced in class as they appeared in the textbook. The analogous problems were then presented and discussed. Students were urged to participate actively in determining and discussing explicitly what was analogous and what was disanalogous between the new problem and the original one and then to make the appropriate changes to the Pascal code.

The word "explicit" is important in this context. It is standard[9] in the context of mathematics and other problem-solving-type instruction to approach new problems by telling students to look back to earlier similar problems (Simon 1980). However, the explicit mappings involved are not ordinarily pointed out. By having students think through and explicitly point out the analogous and disanalogous aspects of problems and the computer programs written

---

[9]See, for example, any college mathematics or introductory physics textbook.

to solve them, I am making make use of the findings of Perkins and Salomon in their comprehensive 1989 paper, "Are Cognitive Skills Context Bound?" After an extensive review of studies of both successful and unsuccessful attempts at teaching thinking skills in the context of a particular academic subject, Perkins and Salomon conclude that any possibility for transfer of skills "depends on learners' deliberate mindful abstraction" of the principles behind the skills.[10] When students are simply told to look back to earlier problems, many students will just follow an earlier problem by rote without taking time to think through the precise similarities and differences. The emphasis on explicit comparisons of similarities and differences should also be looked at as an attempt to activate what Duncker (1945) calls "a genuine thinking process ... characterized by the analysis of the goal, ... of what is demanded, ... by the question 'What do I really want?'"[11] In Kantowski's (1974) study of mathematical problem solving, she expresses the same idea when she states that, "correctly perceiving the elements of a problem, as well as relating these elements to previously acquired knowledge, is essential to finding the solution to a problem."

Appendix E gives the three programming-by-analogy homework-assignment worksheets assigned to students during

---

[10]Emphasis added.

[11]Cited in Kantowski, p. 11.

the term. The worksheets were modeled in a straightforward manner on those used by Grandgenett in his 1989 study of Logo programming instruction and analogical reasoning. Grandgenett's subjects were undergraduate students at Iowa State University enrolled in a course called Educational Applications of Computers.

Grandgenett's worksheets were based directly on Sternberg's (1977) componential analysis of analogical reasoning. (See Section II.1.2.2, above.) The Sternberg analysis has contributed significantly to the study of proportional-analogy type problems. The possible unsuitability of this particular model of analogy for understanding the problem-solving aspect of analogical thinking was not clear to me at the time I designed the worksheets, so I did not at first attempt to deviate from Grandgenett's model.

Appendix A presents one of the worksheets used in Grandgenett's study. The worksheets were used during a three-week unit teaching novice programmers to program in the computer language Logo. The analogical components defined by Sternberg (encoding, inferring, mapping, and applying) are presented in detail on each worksheet, and are an integral part of the students' assignments. As Grandgenett writes, "Using the A:B :: C:D analogy format, the programming activities were structured by the pairing of program output with corresponding program text ... The use

of the Sternberg component processes carefully structured the cognitive tasks of writing the new program."

A comparison of the directions to the student on my Worksheet #1 in Appendix E with Grandgenett's worksheet in Appendix A shows their close similarity. The Sternberg terminology is carried over directly; the only significant change involves the necessary omission of any discussion of graphical output. By the time I was designing Worksheet #3, I realized that the Sternberg terminology was not appropriate for the purposes of this project and eliminated it.

With respect to content, the three worksheets in Appendix E follow the progression of topics that would be expected in an introductory Pascal course. Worksheet #1 presents an exercise in using the simplest type of conditional IF-THEN construction. Worksheet #2 requires use of a sentinel-controlled While loop. Worksheet #3 presents a more complex problem. In order to complete this assignment a student must understand the structural use of procedures and functions in Pascal programs as well as the use of parameters for communication among program components. (It should be noted that in previous semesters when I have taught one-semester introductory programming-language courses, I have been able to include discussion of one-dimensional arrays at the end of the course. The extra time required by the analogical-reasoning material did not

allow for the inclusion of a unit on arrays in this class.)

III.4 Evaluative Instruments

Two types of instruments were administered to this class during the course of the semester: measures of analogical reasoning which were specifically adapted for this project and the Learning Style Inventory, a standardized instrument designed by David Kolb of Case-Western Reserve University "to assess individual orientations toward learning." (See Section II.1.4, above.)

III.4.1 Analogical-Reasoning Measures

III.4.1.1 Overview

The measures used to assess progress in analogical reasoning in this project were adapted from those used by Clement (1994) which, in turn, evolved from those used by Gick and Holyoak in their original 1980 and 1983 studies.

Holyoak (1984) gives a comprehensive description of the development of the original Gick-Holyoak methodology for their work with college students and analogy in the 1980s. A total of eleven different experimental manipulations are presented in the 1980 and 1983 papers. All involved presenting a student with a base analog, most often in narrative form, and observing the student's performance on a target problem. The search for target problems involved consideration of whether a correct solution to a problem could be attained at least as easily by using an analogy with the base problem as by using an alternative problem-

solving strategy. To this end, Gick and Holyoak wanted problems that were to some extent ill-defined so that there would be a choice of possible operations that could be used to solve the target problem. Ten of the 1980/83 protocols use the Duncker radiation problem[12] as the target[13]. The studies examine students' solutions to the radiation problem as a function of different presentations of a base analog, almost always in narrative form.

In the initial protocols in the 1980/83 research, students were presented with a base story about an army general who had to capture a fortress.[14] Students were then asked to solve the radiation problem. In the course of the research, some manipulations presented students with two mutually reinforcing base-analog stories; these manipulations were more effective in producing the desired solution to the target radiation problem than were any of the experiments that presented only a single base analog. Gick and Holyoak (1983) theorize that the greater success is a consequence of a student being able to abstract a schema for the problem when presented with two base stories, both embodying the same principle, rather than with just one base analog. The schema is created from the correspondences

---

[12]See Appendix G, page 10.

[13]The eleventh protocol uses another Gestalt problem, which involves cords used as pendulums (Maier's 1936).

[14]See Appendix G, page 4; the tank commander fills the function of the general.

between the two narratives but exists independently from them; it preserves what the analogs have in common, in particular, it preserves each analog's internal causal relationships and omits their differences. The schema is able to mediate the process of analogical mapping because only the commonalities between the target and the schema need be considered, since any differences between the target and a base analog are not present in the schema.

III.4.1.2 Analogical-Reasoning Measures Used in Present Study

In designing the analogical-reasoning measures to be used in this study, I made use of Clement's recent work, which, while strongly grounded in the analogical-reasoning research paradigms originated by Gick, Holyoak, and Gentner, incorporates innovations to the protocols that have evolved as a consequence of results from relevant research projects that were run in the late '80s and early '90s (Clement 1994). As discussed above, analogical transfer is facilitated when a schema for the base analog has been abstracted. Schemas vary in quality. Recent studies have shown that college students often abstract schemas that mix some surface features of base analogs into the abstract structure of a schema. This tendency has been countered by developing protocols where students are required to represent explicitly the abstract structure common to the analogous situations. These encoding tasks are designed to

force the student to separate a representation of the common relational structure from domain-specific information. These findings are incorporated into the instruments used in this project.

The analogical-reasoning pretest and posttest administered to the Pascal programming students in this study both had the same format but contained different problems and analogies. Both the pretest and the posttest consisted of a 30-minute in-class Learning Session followed in the next class session, two days later, by a 30-minute Problem-Solving Session. The target problem for the pretest was adapted from Clement (1994). The target problem for the posttest is a variation of the Duncker radiation problem. The base narratives were adapted from Gick and Holyoak (1980; 1983); Clement and Gentner (1991); and Clement (1994).

During a Learning Session, each student was given a booklet containing two narratives.[15] As suggested by the Gick/Holyoak work, each story is an instance of the same general problem-solving principle in a different context. The remainder of the protocol then followed Clement's work; students were to engage actively in encoding tasks to aid them in abstracting context-free schemas. Students were asked to read each base story and to describe what they

---

[15]See Appendix F, Part I, for version used in pretest Learning Session; Appendix G, Part I, for version used in posttest Learning Session.

perceived as the problem presented in the story along with the solution strategy the story offered. After they had read and answered the questions about both narratives, students were asked to infer a description of a general problem-solving strategy that would describe the specific strategies used in both stories. Students were not given feedback from the Learning Session.

During a Problem-Solving Session, two days later, students were told that they were being given some problems to solve. They were not told that this exercise was necessarily related to the earlier one. They were given three problems presented in narrative format[16]; only one is a true analog of the stories read two days earlier in the Learning Session; the other two are distractor, or filler, stories. (The use of the distractor problems is discussed in Chapter V.) Students were given seven minutes to work on each problem. For a mapping to be judged successful it had to both access the principle learned in the Learning Session and apply it meaningfully in solving the one problem to which it is applicable in the Problem-Solving Session.

III.4.2 Learning Style Inventory (LSI)

The Learning Style Inventory (Appendix H) was completed by students during the second week of class. The LSI is a twelve-question, sentence-completion, multiple-choice

---

[16]See Appendix F, Part II, for version used in pretest Problem-Solving session; Appendix G, Part II, for version used in posttest Problem-Solving session.

questionnaire designed to measure a respondent's relative emphasis on one of the four complementary modes of learning identified by Kolb. Raw scores for the four different modes are analyzed in combination to construct a snapshot of a student's predominant style of learning at a given point in time. (See Section II.1.4, above.)

Chapter IV discusses outcomes of the procedures described in this chapter.

IV. Project Outcomes

The utility of this exploratory project lies in its bringing together of several diverse areas of inquiry and its generation of investigative questions to be approached in the future. The immediate results are limited. One clear conclusion is that the breadth of the project was too wide, and that further studies must be designed in a modular manner, each one investigating only a single aspect of the issues considered here.

In addition, studies of far transfer effects often yield mixed or negative results (Perkins and Salomon 1987; Holyoak and Koh 1987). The concepts of near and far transfer are attempts to estimate the similarity of the cognitive tasks involved in the performance of task B to those necessary for the performance of task A. For example, the transfer involved in learning to program in Pascal from having learned to program in Basic is clearly an example of near transfer within the domain of programming. On the other hand, an instance where studying mathematics could be shown to improve a person's bridge playing would involve far transfer between two different domains. Nevertheless, as Perkins and Salomon (1987) say, "Often, in teaching thinking, far transfer is the aim because we hope to empower students over a wide range of intellectual challenges."

The remainder of this chapter discusses the results obtained and suggests changes to be made in the design and

implementation of future studies.

IV.1 Analogical-Reasoning Measures

As discussed in Chapter III the analogical-reasoning measures used in this study were adapted from those used by Clement (1994) and follow from the prototypes used by Gick and Holyoak (1980, 1983) and subsequent researchers in the field. These models all present a base-analog story (or stories) followed by a target story to which the student is supposed to map the base analog.

IV.1.1 Pretests

The materials presented to the students as pretests appear in Appendix F. The pretests were administered on February 1 and 3, 1994, before any instruction on programming by analogy was given in class.

IV.1.1.1 Pretest Learning Session

The pretest Learning Session (Appendix F, Part I) consists of two base-analog stories from which students are to abstract a common schema. As discussed in Chapter III, the technique of providing two base stories has been found to reinforce the abstraction of the schema that the two stories have in common (Gick and Holyoak 1983). It is a method of helping students distinguish the abstract concept that the stories share from the surface features of each story.

The base stories in Appendix F were developed, pilot tested, and used in research with college students by

Clement (1994). They are designed to present the abstract concept that Clement refers to as the "mimic paradigm": An entity needs to cope with a dangerous situation; in the process of interacting with this dangerous situation, the entity could be destroyed. The solution is to develop coping abilities beforehand by interacting with a harmless mimic of the dangerous situation.

Story 1, the first instance of the mimic paradigm in the pretest Learning Session, describes how army recruits are exposed to the trappings of war in a nonlethal setting in bootcamp in order to help them develop the ability to cope with dangerous real-war situations. Story 2 describes how space ship material is prepared to endure exposure to a dangerous gas by first being exposed to a similar nonharmful gas. In the Learning Session, students were asked to express in their own words what they perceived to be the problem posed and the strategy used to solve the problem in each story. Then, after they had read both stories, in an attempt to aid in the abstraction of the common schema, students were asked to identify the similarities they saw between the two stories with respect to the strategies used to solve the problems posed in each.

Student responses to the pretest Learning Session are given in Table III. The responses are analyzed in terms of whether or not a student identified (1) the problem in the story; (2) the strategy used to solve the problem; and (3)

the schema embodied in both stories. For Story 1, of the eight students whose responses are given in the table, four identified the problem, three gave partial identifications, and one did not identify the problem. Six of the students identified the strategy for Story 1 (all who had identified the problem and two who had given partial identifications), and two did not identify the strategy. For Story 2, six identified the problem, one gave a partial identification, and one did not identify it. The strategy for Story 2 was identified by three students (all of whom had identified the problem), partially identified by three, and not identified by two. Four of the students abstracted the schema (one extremely well); and four did not.

IV.1.1.2 Pretest Analogy Session

Appendix F, Part II, presents the target story and the two accompanying distractor stories that were used in the pretest Analogy Session, which was administered two days after the pretest Learning Session. The use of distractor stories in the Analogy Session follows Clement's design. They are included to increase the difficulty of accessing the problem solving schema. They are stories that describe dilemmas that need to be solved, but are deliberately designed not to be solved by the problem solving schema taught in the Learning Session. The pretest target story is an instance of the mimic schema embedded in a story about fictitious marine creatures called drats and fictitious

amphibians called zylots.

Student responses to the pretest Analogy Session are given in Table IV. Three of the students used the mimic schema in solving this problem; three of the students made partial use of the schema; and four did not use it at all.

IV.1.2 Posttests

Appendix G comprises the posttest materials. The posttest was administered on April 26 and 28, 1994, during the last week of classes for the Spring 1994 semester. By that time students had been exposed to in-class discussion of analogy as a way of thinking (Appendix I and short discussions in several class sessions) and as a way of writing computer programs (Appendix D and other similar examples). They had also written three computer programs that were presented to them using the programming by analogy worksheets in Appendix E.

IV.1.2.1 Posttest Learning Session

The posttest Learning Session (Appendix G, Part I) presents students with two base narratives that evolved from the Gick/Holyoak research on Duncker's Radiation Problem. They are instances of what is known as the convergence paradigm: A centrally located target must be destroyed. A strong force is required, but constraints within the problem preclude force being applied along a single path. The solution is to apply several weak forces along many paths simultaneously.

In the posttest Learning Session, Story 1 is about fighting a fire when there is no hose large enough available to deliver sufficient foam to the fire all at once. Story 2 presents a situation where in order to capture an island a large military force must be broken up and simultaneously sent down several narrow bridges.

Student responses to the posttest Learning Session are given in Table V. For Story 1, six students identified the problem, three students made partial identifications, and one student omitted this question. Seven students identified the strategy for the problem in Story 1 (four who had identified the problem, two who had made partial identifications, along with the student who had omitted the identification of the problem); three students made partial identifications. Four students identified the problem in Story 2; three made partial identifications; and three did not identify the problem. The strategy for Story 2 was identified by six students (four who had identified the problem, one who had made a partial identification, and one who had not identified the problem); three students made partial identifications, and the answer that one student gave was indecipherable. Four students correctly abstracted the schema, four gave partially correct versions, and two did not abstract the schema at all.

IV.1.2.2 Posttest Analogy Session

Appendix G, Part II, presents the target story and two

accompanying distractor stories that were used in the posttest Analogy Session, administered two days after the posttest Learning Session. The posttest target story is a version of the Duncker radiation problem, the original instance of the convergence schema.

Student responses to the posttest Analogy Session are given in Table VI. None of the ten students used the convergence paradigm for the radiation problem. This result, along with analysis of the results tabulated in Tables III, IV, and V, is discussed below.

IV.2 Analysis

The data in Tables III through VI must be looked at across all pretests and posttests. In the pretests (Tables III and IV), of the three students who applied the mimic schema in solving the target (zylots) problem, two had correctly identified the problems, strategies, and schema in the Learning Session, but one had missed the Learning Session. This student was clearly making independent use of the mimic schema. Of the three students who made partial use of the mimic schema in solving the target problem, one had not identified the problems, strategies, or schema in the Learning Session. This student may also have been making independent use of the schema, although there may have been some residual affect from the Learning Session even though he could not articulate and understanding of the schema.

The posttests (Tables V and VI) produced a completely unexpected result: None of the students made use of the convergence schema in solving the radiation problem. This was true even of the two students who identified both problems and their solution strategies and also abstracted the convergence schema in the Learning Session. (These two students had also identified both problems, strategies and the schema in the pretest Learning Session and had used the schema in solving the pretest analogy.)

This result is particularly surprising because in Gick's and Holyoak's work (1980, 1983), they found that on average 10% of their participants employed the convergence schema in solving the radiation problem prior to being exposed to any experimental treatment. Thus, a negative transfer effect may have occurred. I see this as an issue for further investigation.

IV.3 Learning Style Inventory

Experiential learning theory predicts that computer scientists will on the whole exhibit the characteristics of convergent learning: grasping knowledge through abstract conceptualization and transforming it by active experimentation. The results of the administration of the Learning Style Inventory for this project during the second week of class are presented in Table VIII. No conclusions can be drawn from this small sample. In fact, the four students who received As in the course represent all four of

Kolb's learning styles: one diverger, one assimilator, one converger, and one accommodator. The only expected result that can be pointed to is that the A student who is a chemistry major strongly showed convergence as a learning style. (Chemists, too, are predicted, on the whole, to be convergers.) Nevertheless, I believe that further study in this area is important. The percentage of students lost from introductory programming courses is unacceptably high, and all avenues of addressing the issue must be pursued. At the present time the Department of Computer Science at the U.S. Air Force Academy is experimenting with assigning students in their introductory programming classes to instructors who exhibit the same learning style on the LSI.[17] I am anxious to continue research on this issue.

---

[17]Personal communication.

V. Plans for Future Research

The goal of this exploratory project is the formulation of research questions and an experimental method to be used in a formal controlled study designed to address the following issues: Is it possible to integrate the teaching of analogical reasoning skills into an introductory computer programming course so that (1) students develop skill in analogical reasoning; (2) the analogical reasoning component of the course enhances the quality of learning of the computer skills; and (3) the analogical reasoning component does not significantly diminish the quantity of computer-skills-related material that is learned?

V.1 Initial Preparation

Preparation for the formal experiment will require approximately one year (see Appendix K, "Time Line for Study") and will encompass the following activities: (1) location of an appropriate site and instructors; (2) continued review of the instructional protocols, including teaching methodology and design of programming worksheets, revision and pilot testing of the analogical reasoning measuring instruments, and analysis of the use of the Learning Style Inventory; and (3) training of instructors.

V.2 Logistical Planning

Ideally, the study will involve the participation of four sections of an introductory computer programming course, two to be taught as experimental sections and two as

control sections. Thus, it will first be necessary to locate an appropriate site. Appropriateness requires the presence of at least four sections of an introductory programming language course, departmental interest in experimental teaching methodologies and willingness to cooperate with the project, and the presence of course instructors interested in the project's hypotheses.

To increase the probability of obtaining significant results, the course instructors selected will be people who have both experience teaching the introductory programming language course and a concern for students' developing a conceptual understanding of computer science. The instructors should also be interested in developing teaching techniques appropriate for students with a variety of learning styles, not only for those who naturally perform well in traditionally structured computer science classes. In sum, the instructors should be actively interested in the project's hypotheses while at the same time not heavily invested in them.

The instructors must be willing to spend approximately two weeks before the semester begins in training sessions (see Section V.4, below) where they will be introduced to the course sequence, course materials, and teaching methodologies. Each instructor will teach one experimental section and one control section. They will agree to administer the measuring instruments in both of their

classes and to teach one of their sections using the programming by analogy method. They will be provided with all materials necessary for the programming by analogy sections, including examples to present in class, assignments, and exams.

V.3 Instructional Protocols

Review of the instructional protocols of the exploratory study will concentrate on analyzing their deficiencies and designing methods to improve them. The analysis will be framed by an attempt to understand why transfer proved so difficult.

V.3.1 Teaching Methodology

As Perkins and Salomon (1987) state in their article Transfer and Teaching Thinking, transfer of intellectual skills, while always difficult, proceeds, when it does occur, "through the mindful abstraction and application of principles learned." They continue, "The abstraction required ... entails genuine understanding of the principle and its relation to particular cases, and a grasp of the conditions under which the principle might apply." Since the exploratory study did not give evidence of transfer, questions that must be addressed are (1) How might the controlled study provide additional explicit opportunities for students to abstract the concepts involved in analogical reasoning? and (2) How might the protocol provide students with a wide variety of learning situations in which to

practice the concepts that they are learning? To this end, the following changes in methodology will be considered: telling students at the beginning of the semester that they are involved in a project that is studying reasoning by analogy, thus making the topic of analogy explicit; and, as means of reinforcing attention to analogical reasoning, reminding students throughout the semester that part of their course work involves learning how to apply analogies appropriately, and placing one or two questions that test analogical reasoning on each class exam. The overall intention of these changes is to better provide what Perkins and Salomon refer to as the type of "scaffolding for learning" that they have found to aid students in abstracting principles and transferring them from one learning situation to another.

V.3.2 Programming Worksheets

V.3.2.1 Worksheet Design

The programming worksheets used (Appendix E) were modeled on Grandgenett's (Appendix A), which were geared towards proportional-analogy-type[18] test problems. His worksheet's were developed using Sternberg's model of analogy. (See II.1.2.2, above.) Further research has shown me that, as Holyoak (1984) says, "analogical problem solving

---

[18]Proportional analogies are well known; while there are several types, two that are commonly used are verbal analogies (such as attorney : law :: physician : medicine) and geometrical analogies (such as ► : ◄ :: → : ←).

invokes mental processes that go beyond those postulated by models [such as Sternberg's] of solving [proportional] analogy problems." While it is possible to force a base-target-type analogy that may be useful in problem solving into a proportional-analogy-type format, the procedure is unnatural and cumbersome. Since the analogical reasoning worksheets used in this exploratory project did not have the hoped for impact on students' abilities to recognize and use analogies, the differences between the processes involved in analyzing proportional analogy problems and those necessary for using analogies in problem solving are relevant.[19]

Perhaps the most important difference between the two ways of using analogy is that a proportional-analogy problem is to be solved for its own sake. It is a given, presented to the potential solver in its entirety. On the other hand, the essence of the use of an analogy in problem solving, whether the writing of a computer program, the construction of a geometry proof, or the design of an architectural drawing, involves, first, the noticing of a possible base problem and, subsequently, the analysis (not necessarily conscious) of its potential usefulness for a given target problem. Moreover, the essence of the suitability analysis necessary for successful analogical problem solving is its

---

[19]The widespread use of proportional-analogy-type problems on computer-programming aptitude tests is another reason for examining this difference.

open-endedness. Often, possible analogies come to mind and are rejected when the disanalogous aspects of the potential base and target outweigh the analogous aspects. Proportional analogy problems are designed so that one of the multiple-choice answers always properly completes the analogy. So, while there is some evaluation involved, it is not an open-ended thought process. Finally, solving a proportional analogy is not an integral part of a more complex, directed thought process.[20]

Revision of the programming worksheet format, is, therefore, warranted. Appendix J provides a redesigned worksheet. To aid in the mapping of the solution schema for the programming problem being assigned, the revised worksheet presents two base problems that have the same programming structure. As discussed above, mutual reinforcement of the same problem-solving schema by two different base problems has been found to aid significantly in the abstraction of the schema and its mapping to the target problem, so this seems a crucial change to incorporate in the worksheet design in the effort to increase the likelihood of transfer in the formal study.

V.3.2.2 Worksheet Administration

Student Response. The instructors in the formal study will be encouraged to discuss the nature of the worksheets

---

[20]This discussion of the difference between proportional analogy type problems and the use of analogy in problem solving is based on material summarized by Holyoak (1984).

with the students. In particular, students who feel that the worksheets might limit the expression of their creativity in programming will be reassured that any additional programs that they write on their own will be valued and taken into account in their overall class evaluation. It is unlikely that many students in a typical introductory programming class will have this reaction. In my experience, most introductory students are more than happy with any assistance they are given in learning how to program. However, it is important that any students who do find the worksheets constraining be encouraged to do other work on their own.

Frequency of Administration. In the exploratory project, students were assigned only three programming worksheets throughout the semester. (See Appendix E.) As described in Chapter III, two of the worksheets contain relatively simple programming problems, while the problem on the third worksheet is considerably more difficult. It is likely that this lack of sufficient reinforcement of the active use of analogies contributed to the lack of transfer observed in the exploratory study. Therefore, for the formal study, the protocol will be changed to provide a minimum of five worksheets, each one dealing with subject matter covered in the previous two- to three-week period. (See Appendix K.) Instructors will be urged to return the corrected worksheets to students within a week, and to

devote part of a class meeting to discussion and review of both the computer programming topics dealt with on the worksheet and the analogical reasoning skills that were applied (or misapplied) by students as they completed the worksheet. The goal of this process is the encouragement of students to think actively about analogy.

V.3.3 Measuring Instruments

V.3.3.1 Analogical Reasoning Measures

The analogical-reasoning measuring instruments used in the exploratory study (Appendices F and G) will be carefully reviewed and revised. Pilot testing of other possible analogical reasoning narratives will be conducted with college students who are enrolled in introductory programming courses. One area that needs to be addressed by the pilot testing is the suitability of the particular stories used in the analogical mapping exercises in the exploratory study. Work by Clement and Gentner (1991) deals with the problem of student subjects' responses to analogical reasoning exercises possibly being governed by prior knowledge and beliefs about the source and target domains. They have, therefore, developed analogy exercises where the source and target are totally novel fictional domains. (See Appendix L.) While the zylots-drats mimic-paradigm exercise (Appendix F) does take place in a fictional domain, both the mimic paradigm and the convergence paradigm for the radiation problem have strong

resonances with real-world medical situations. Thus, lack of transfer in the pilot study may have resulted from students' employing their prior knowledge to reason about the analogical exercises rather than the material presented in the passages presented to them. Thus, it makes sense to pilot test the Clement-Gentner analogy exercises that have been carefully constructed to cut down significantly on interference from real-world "noise" in the mapping of analogical problem solutions. The pilot tests will examine students' responses to the analogical situations presented to them both before any discussion of analogy and after they have been given a hint[21] to look for analogies between the source and target problems. This process demonstrates if there is improvement in mapping after a hint is given. If improvement is found, the experimental situation designed for the formal study will be constructed to test if similar improvement in mapping occurs when students are exposed to instruction in programming by analogy.

The use of distractor stories will also be investigated in the pilot studies. Since even students who identified the convergence schema in the posttest Learning Session had difficulty mapping the schema to the radiation problem, the introduction of additional difficulty with the distractor problems must be reexamined.

---

[21]Much of the Gick-Holyoak (1980; 1983) work studies the difference between student responses to analogical exercises before and after being given a hint to look for analogy.

V.3.3.2 Learning Style Inventory

I see continued research on the use of the Learning Style Inventory in introductory computer programming classes as an integral component of the formal experiment. I believe that the underlying concepts of experiential learning, as described in Section II.1.4, above, have much to contribute to improvement in both the quality of learning and student retention rates in introductory programming courses. As discussed in Section IV.3, above, a large number of students who attempt to study computer programming become discouraged in the first formal class in programming in which they enroll. We lose many students with potential for success in computer science because not enough attention is given to what they bring to the entry level programming class in terms of both variety of learning styles and nontraditional backgrounds. These students will be unnecessarily handicapped if they are not given the opportunity to master the fundamentals of computer programming in a procedural language.

While there is considerable discussion in computer education circles as to whether, in this age of the ever increasing use of prepackaged software, it is still important for students who do not intend to study computer science on a more advanced level to learn how to program, I think that a strong case can be made for an affirmative answer to that question. I believe that once students

understand how a program is written, they can develop a much more sophisticated sense of what is taking place inside the computer when they are using software packages. Thus, they have a much stronger base on which to build their knowledge of increasingly sophisticated software packages.

As discussed in Section IV.3, above, the results of the administration of the LSI in the exploratory study were inconclusive. A formal controlled study that involves fifty or more students, however, may be expected to yield both more data and data of greater significance.

V.3.3.3 Statistical Measures

Discussion, evaluation, and pilot testing of possible statistical measures to be included in the formal study will be an integral part of the review process. Measures must be developed for use both within and between experimental classes and control classes with respect to performance in development of computer programming skills and performance on analogical-reasoning measures. This work will be undertaken in consultation with an experienced statistician.

Of primary concern is the development of appropriate, reliable measures to compare student improvement in analogical-mapping skills between the pretest and posttest analogical-reasoning measures in both the experimental and control classes. Analysis, will, of course, study whether one particular learning style predominates in those students who are most successful in their course work and/or their

understanding of the use of analogies. But of equal interest will be a search for patterns involving students in the experimental and control classes. Some of the questions to be asked: Do students who favor a particular learning style tend to learn computer programming better in a conventionally taught computer programming course? How do learning style and development of ability in analogical reasoning interact? More fine-grained analyses may study how a student's position on the learning style grid, i.e., close to the axes or deeper into a quadrant (see Figure 3), affects learning in the experimental and control classes.

Measuring instruments used by Fay (1990) and Grandgenett (1989) in their studies of computer programming instruction and analogical reasoning will also be evaluated for possible adaptation. Fay's instruments include measures of progress in computer programming. She runs correlations of numbers of programs completed and program quality between students in treatment and control classes. Program quality measures include closeness of match of program output to program specifications, modularity in program construction, local efficiency (i.e., proficient use of individual commands), and global efficiency (i.e., accurate use of local and global parameters, re-use of subprocedures).

Both Fay and Grandgenett find summary statistics for responses to the items on their student background questionnaires (similar to the one in Appendix C). They

calculate means, standard deviations, and F-values between experimental and control groups for age, computer nervousness, grade-point average, years of previous mathematics and computer science instruction, and year in college. Grandgenett does detailed correlational analyses of mean performance on computer programming and analogical reasoning measures for experimental and control groups while controlling for these variables from the Student Background Questionnaire. Careful evaluation of the appropriateness of the protocols used in these two studies will be done during the period of review and pilot testing before decisions are made as to just which statistical measures are suitable for the formal study.

Appropriate statistical analyses in a controlled study will provide significant information as to the feasibility of the effective teaching of analogical reasoning within an introductory computer programming class.

V.4 Training of Instructors

As discussed in Section V.2, ideally, the instructors will be people with some teaching experience and an active interest in and openness to the development of innovative, effective teaching strategies for the introductory computer programming course. They may be graduate students. It is not expected that they have familiarity either with issues concerning the teaching of abstract thinking to college students or with Kolb's experiential learning theory.

The pre-semester training sessions will total about twenty hours over a two-week period some time during the six weeks prior to the beginning of the school semester. (See Appendix K.) I hope to have grant money to pay the instructors a small stipend for this extra preparation time. I will, of course, also be available to them throughout the teaching semester to respond to questions that may arise and to deal with unforeseen complications that may develop.

The training will focus on those areas of the project with which the instructors will probably not have had experience. They will be asked to read this dissertation several short articles, and selected passages from David Kolb's book, Experiential Learning (1984). Training sessions will be conducted in a dialogic manner, with every attempt made to involve the participants in the discussions and to elicit questions and topics for further discussion from them.

Specific theoretical topics to be covered in the training sessions include: analogical reasoning as a form of abstract thinking; the ways in which expert programmers make use of analogies in writing programs; the widespread deficiencies in abstract-thinking skills demonstrated by many college students; and fluidity of learning style and learning as process as integral parts of Kolb's experiential learning theory, including their effects on classroom teaching.

On a pragmatic level, there will be detailed discussions of all of the instruments that will be administered: the background questionnaire, the Learning Style Inventory, and the analogical reasoning pretest and posttest. Discussions of the pretest and posttest will center on those in Appendices F and G; the instructors will not be shown the passages chosen after pilot testing specifically for use in the formal study.

Additional discussions will deal with pragmatic concerns, including the introductory lecture on analogical reasoning (Appendix I), the classroom examples (Appendix D), and revised programming worksheets (Appendix J). The floor will be open for any other issues that the instructors may wish to discuss. My goal is for the training experience to set the stage for a productive, supportive teaching experience.

Figures

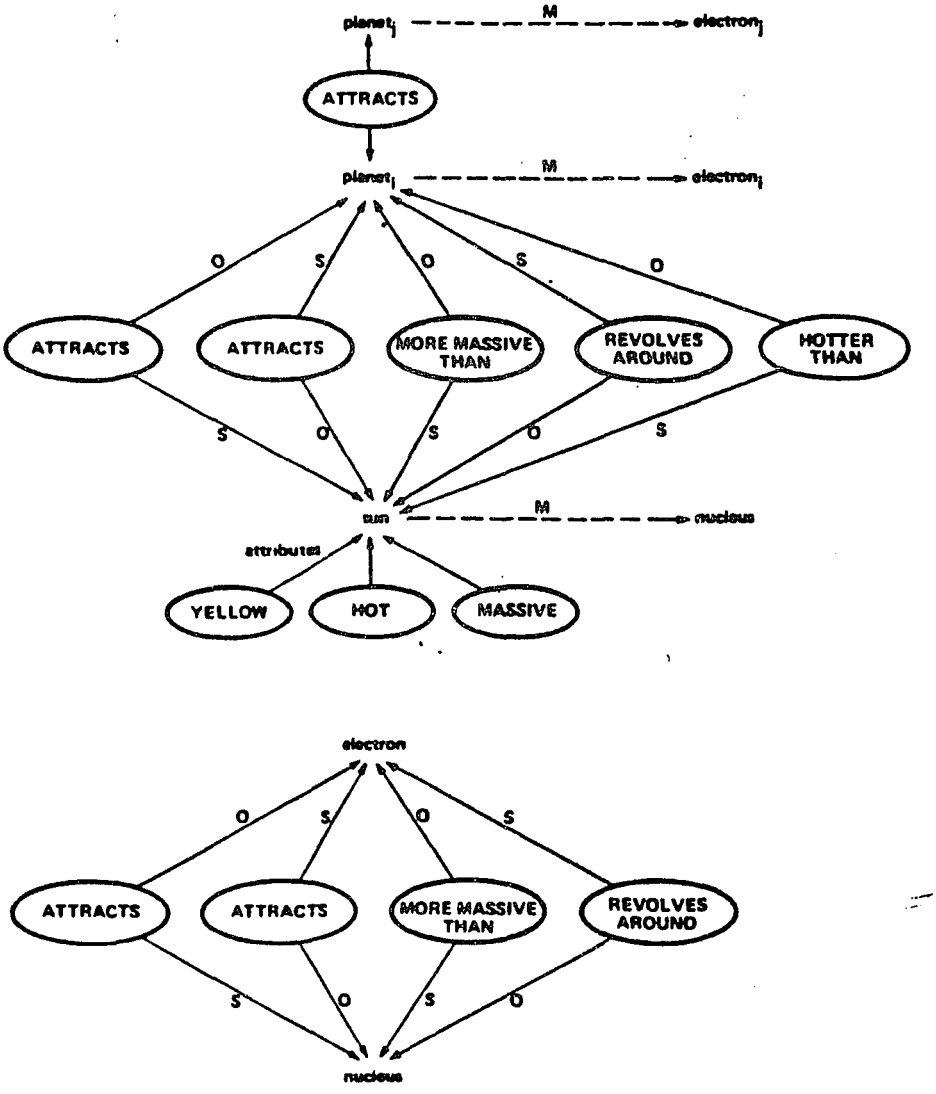


Figure 1. Structure-mapping diagram for the analogy: "The atom is like the solar system."

(Adapted from Gentner (1983), page 160).

M = "is mapped to"

s = subject

o = object

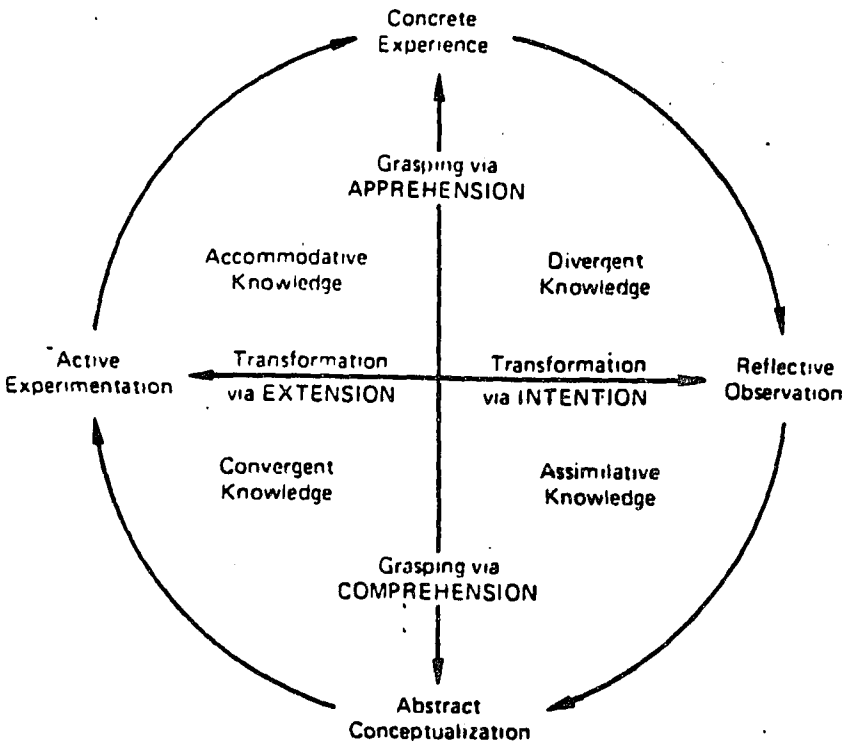


Figure 2. The underlying structural dimensions of the learning process in Kolb's experiential learning theory.

(Adapted from Kolb (1984), page 42.)

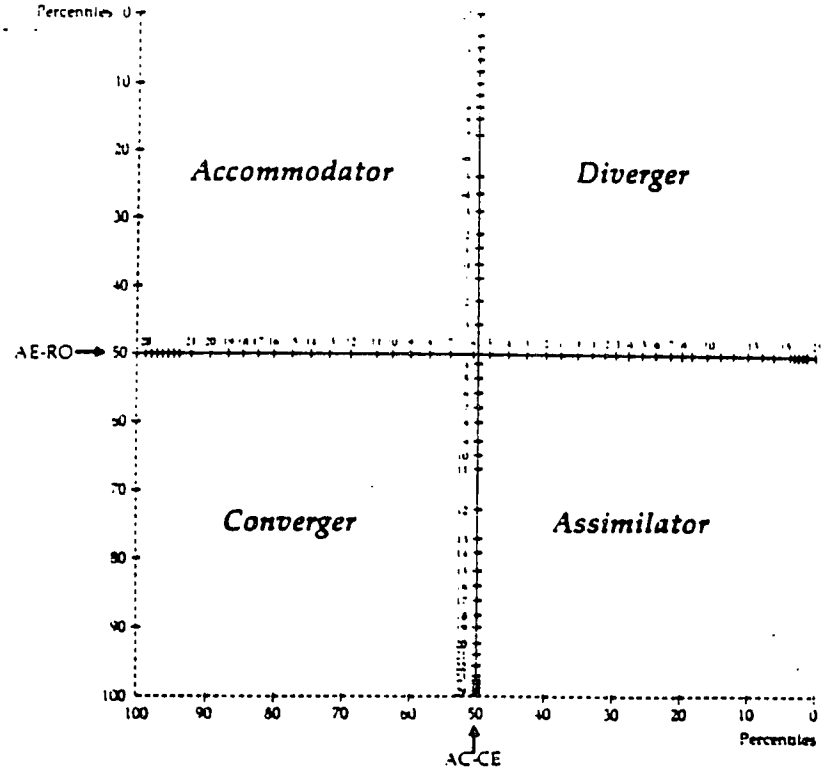


Figure 3. The learning-style type grid.

(Adapted from McBer & Co., Learning Style Inventory, pg 6.)

Tables

Table I. School-Related Data from Student Background Questionnaire

STUDENT CODE	ACADEMIC YEAR	MAJOR	CAREER GOAL	GPA
ABO 05	N/A[3]	N/A	N/A	N/A
ALL 05	2	Electrical Engineering	Undecided	3.0-3.49
ALL 14	N/A	N/A	N/A	N/A
ANT 23	1	CIS/Math	Computers	2.5-2.99
AUG 04	2	CIS	Computers	2.0-2.49
BOS 25[1]	1	Business	Business	N/A
DAN 02	2	CIS	Computers	N/A
EFU 26[1]	2	Accounting/ CIS	Sales	3.5-4.0
JAB 21	2	CIS	Undecided	2.5-2.99
JAK 13[1]	4	Chemistry/ Math	Chemistry	3.5-4.0
JEA 18[1]	1	Undecided	Undecided	3.0-3.49
JER 08[1]	3	CIS	Undecided	3.0-3.49
MEL 12	1	CIS	Undecided	N/A
NMI 07[1]	1	CIS	Computers	3.0-3.49
NMI 24	N/A	N/A	N/A	N/A
NMI 28[1]	2	CIS	Typesetting	3.5-4.0
OBD 05	N/A	N/A	N/A	N/A
PRI 19[1]	2	CIS	Undecided	2.5-2.99
QUE 22	1	CIS	Undecided	1.5-1.99
RHK 05	N/A	N/A	N/A	N/A
ROB 03[2]	2	CIS	Computers	2.0-2.49
THO 24[2]	2	CIS	Computers	3.0-3.49

[1]Student participated in all activities.
[2]Student was absent for Learning Pretest.
[3]The students for whom N/A (not available) appears in all columns except the first were absent on the date on which the student background questionnaire was completed.

Table II. Personal Data from Student Background Questionnaire

STUD CODE	COMPUTER ANXIETY	GENDER	BIRTH YEAR	ETHNIC GROUP	FIRST LANG	CURR LANG
ABO 05	N/A[3]	N/A	N/A	N/A	N/A	N/A
ALL 05	somewhat conf	M	74	BLACK	ENG	ENG
ALL 14	N/A	N/A	N/A	N/A	N/A	N/A
ANT 23	somewhat conf	M	71	BLACK	ENG	ENG
AUG 04	somewhat conf	F	74	BLACK	ENG	ENG
BOS 25[1]	somewhat conf	M	74	BLACK	GHAN[4]	ENG
DAN 02	very conf	M	71	BLACK	ENG	ENG/ GHAN[4]
EFU 26[1]	not nerv not conf	F	73	BLACK	FANTI[5]	ENG/ FANTI
JAB 21	somewhat conf	M	74	BLACK	ENG	ENG
JAK 13[1]	very conf	F	72	WHITE	RUSS	RUSS[6]
JEA 18[1]	not nerv not conf	F	64	BLACK	ENG	ENG
JER 08[1]	somewhat conf	M	73	BLACK	ENG	ENG
MEL 12	very conf	M	75	BLACK	ENG	ENG
NMI 07[1]	very conf	M	75	WHITE	ENG	ENG/ MACED[7]
NMI 24	N/A	N/A	N/A	N/A	N/A	N/A
NMI 28[1]	very conf	M	71	WHITE	ENG	ENG
OBD 05	N/A	N/A	N/A	N/A	N/A	N/A
PRI 19[1]	somewhat conf	M	74	BLACK	ENG	ENG

QUE 22	not nerv not conf	M	75	BLACK	ENG	ENG
RHK 05	N/A	N/A	N/A	N/A	N/A	N/A
ROB 03[2]	somewhat conf	M	74	HISPAN	SPAN/ ENG	SPAN/ ENG
THO 24[2]	not nerv not conf	M	72	WHITE	ENG	ENG

[1]Student participated in all activities.
[2]Student participated in all activities except for Learning Pretest.
[3]The students for whom N/A (not available) appears in all columns except the first were absent on the date on which the student background questionnaire was completed.
Ghanaian dialect.
[5]A Ghanaian dialect.
[6]Russian.
[7]Macedonian.

Table III. Learning Pretest
(Appendix F--Part I)
Student Responses[1]

STUDENT CODE	Story 1 Ident[3] problem	Story 1 Ident[3] strategy	Story 2 Ident[3] problem	Story 2 Ident[3] strategy	Abstracts schema
BOS 25	Yes	Yes	Yes	Partial	No
EFU 26	No	No	Partial	No	No
JAK 13	Partial	Yes	Yes	Partial	Yes
JEA 18	Yes	Yes	Yes	Yes	Yes
JER 08	Partial	Yes	Yes	Partial	Yes
NMI 07	Yes	Yes	Yes	Yes	No
NMI 28	Yes	Yes	Yes	Yes	Yes[4]
PRI 19	Partial	No	No	No	No
ROB 03[2]	-------	-------	------	-------	-------
THO 24[2]	-------	-------	------	-------	-------

[1]The data in Tables III through VIII refer to the eight students for whom data is complete, and the two students who are missing only the Learning Pretest.
[2]Not present on day Learning Pretest was administered.
[3]Identified.
[4]Very accurate identification of principle.

Table IV. Analogies Pretest
(Appendix F--Part II)
Student Responses[1]

STUDENT CODE	Use of Mimic Schema in Target Problem
BOS 25	Partial
EFU 26	No
JAK 13	No
JEA 18	Yes
JER 08	No?
NMI 07	No
NMI 28	No
PRI 19	Partial
ROB 03	No
THO 24	Yes[2]

[1]The data in Tables III through VIII refer to the eight students for whom data is complete, and the two students who are missing only the Learning Pretest.
[2]This student missed the Learning Pretest.

Table V. Learning Posttest
(Appendix G--Part I)
Student Responses[1]

STUDENT CODE	Story 1 Ident problem	Story 1 Ident strategy	Story 2 Ident problem	Story 2 Ident strategy	Abstracts schema
BOS 25	Yes	Yes	Yes	Yes	Yes
EFU 26	Partial	Yes	Partial	Yes	No
JAK 13	No Ans	Yes	Partial	Not Clear	Partial
JEA 18	Yes	Yes	Yes	Yes	Yes
JER 08	Yes	Partial	Yes	Yes	Yes
NMI 07	Yes	Partial	Partial	Partial	Partial
NMI 28	Yes	Yes	Yes	Yes	Yes
PRI 19	Partial	Partial	No	Partial	No
ROB 03	Yes	Yes	No	Partial	Partial
THO 24	Partial	Yes	No	Yes	Partial

[1]The data in Tables III through VIII refer to the eight students for whom data is complete, and the two students who are missing only the Learning Pretest.

Table VI. Analogies Posttest
(Appendix G--Part II)
Student Responses[1]

STUDENT CODE	Use of Convergence Schema in Target Problem
BOS 25	No
EFU 26	No
JAK 13	No
JEA 18	No
JER 08	No
NMI 07	No
NMI 28	No
PRI 19	No
ROB 03	No
THO 24	No

[1]The data in Tables III through VIII refer to the eight students for whom data is complete, and the two students who are missing only the Learning Pretest.

Table VII. Student Grades[1]

STUDENT CODE	EXAM 1	EXAM 2	FINAL EXAM	FINAL GRADE
BOS 25	77	70	73	C+
EFU 26	68	81	75	B-
JAK 13	93	95	100	A
JEA 18	88	85	100	A
JER 08	90	83	99	A
NMI 07	88	88	86	B
NMI 28	87	100	100	A
PRI 19	32	56	55	D
ROB 03	46	46	56	D
THO 24	76	93	73	B

[1]The data in Tables III through VIII refer to the eight students for whom data is complete, and the two students who are missing only the Learning Pretest.

Table VIII. Learning Style Inventory[1]

STUDENT CODE	LEARNING STYLE
BOS 25	Assimilator
EFU 26	Diverger
JAK 13	Converger
JEA 18	Diverger
JER 8	Assimilator
NMI 7	Converger
NMI 28	Accommodator
PRI 19	Assimilator
ROB 3	Diverger
THO 24	Accommodator

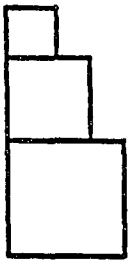
[1]The data in Tables III through VIII refer to the eight students for whom data is complete, and the two students who are missing only the Learning Pretest.

Appendices

APPENDIX A
GRANDGENETT SAMPLE ANALOGICAL REASONING WORKSHEET[1]

Analogical Reasoning Sheet

Graphic *(previous graphic output)*



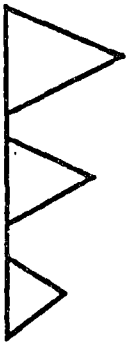
Code *(previously designed code)*

```
To Stack :X
Square :X
Fd :X
Square :X-10
Fd :X-10
Square :X-20
End
```

```
To Square :X
Repeat 4 [Fd :X Rt 90]
End
```

Sheet # SA3

Graphic *(graphic output desired)*



Code *(modified code needed)*

Directions:

1) Sketch the graphical output desired. *(defining the problem)*
2) Let's look at a previous problem to help us. *(choosing a plan)*
3) Now employ our steps for analogical reasoning. *(carrying out the plan)*

- **Encode:** Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.
- **Infer:** Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).
- **Map:** Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.
- **Apply:** Now, using what you can from the previous problem, write a program to produce the desired graphical output.

4) How did the program work? Describe briefly below: *(looking back)*

Desired Output? Yes No Not Quite

What Seems Wrong? describe in words

Part to Modify: describe in words

No sweat! I'll Just Try Again!

[1]Grandgenett (1989), p.205.

Appendix B
Student Participation Form

I am requesting your help with a research project this semester. The project is studying computer science teaching. Your participation will be extremely helpful and will be very much appreciated.

You will be asked to do the following during the course of the semester:

(1) Fill out several questionnaires.
(2) Read several passages and respond to questions about them.

THE DATA COLLECTED FOR THIS PROJECT WILL HAVE NO BEARING ON YOUR GRADE FOR THIS COURSE.

The following steps are being taken to ensure the confidentiality of your responses:

(1) A coding system will be used so that your name will not appear on any of the material to which you respond.
(2) The results of the study will be reported in a manner in which it will be impossible to identify any individual participant.

****

Date:_____

I understand the participation being requested of me in the research project that will be conducted in CIS 102 this semester. I further understand that my responses to questions will be completely confidential and that a coding system will be used to protect my confidentiality. I also understand that my participation in this project will have NO bearing on the grading of this course.

_____
(please print your name)

_____
(please sign your name)

****

Code: ___ __

Both to preserve your anonymity and to enable the researcher to match this form with other forms that you will be completing, please enter the following code in the space marked Code above:

The first three letters of your middle name followed by the day of the month of your birthday. (For example, if your middle name were Emily and your birthday September 24, you would enter: EMI 24.)

If you have only a middle initial, please enter the initial followed by ON. If you have no middle name, please enter NMI; if the day of your birthday is a single digit, please enter 0 before the digit.

Appendix C
Student Background Questionnaire

a. Both to preserve your anonymity and to enable the researcher to match this form with other forms that you will be completing, please enter the following code in the upper-right-hand corner:

The first three letters of your middle name followed by the day of the month of your birthday. (For example, if your middle name were Emily and your birthday September 24, you would enter EMI 24.)

If you have only a middle initial, please enter the initial followed by ON. If you have no middle name or initial, please enter NMI. If the day of your birthday is a single digit, please enter 0 before the digit.

b. Please answer all questions to the best of your ability.

****

I. General Academic Information

1. Academic year: Freshman Sophomore Junior Senior

2. Major Department, if you have one:

3. If you have a career goal at this time, please describe it:

4. Why are you taking this course?

5. Please place a check beside your current GPA:

(a) 4.0 to 3.5 ___ (d) 2.49 to 2.0 ___
(b) 3.49 to 3.0 ___ (e) 1.99 to 1.5 ___
(c) 2.99 to 2.5 ___ (f) Below 1.5 ___
(g) First semester freshman, no college GPA ___

II. Math/Computer Science Background

(If the answer to any of these questions is None, please write None.)

6. Please list all of the math courses that you took in high school:

Code: ___ __

7. Please list all of the math courses that you have previously taken at any college:

8. Please list all of the math courses that you are taking this semester:

9. Please list all of the computer courses that you took in high school:

10. Please list all of the computer courses that you have taken at a job:

11. Please list all of the computer courses that you have taken at any college:

12. Please list any other computer courses that you are taking this semester:

13. If you have had any experience working with computers, please describe it briefly:

14. If you have ever written any computer programs, please list the languages that you used:

Code ___ __

15. Please check the way you would describe how you currently feel about computers:

(a) very nervous ___
(b) somewhat nervous ___
(c) not really nervous, but not really confident either ___
(d) somewhat confident ___
(e) very confident ___

III. **Demographic Information**

16. ___ female ___ male

17. ___ single ___ married ___separated/divorced___widowed

18. _____ year of birth

19. Which one of the following do you consider to be your ethnic/racial group? (Check one only.)

(a) Asian ___
(b) Black ___
(c) Hispanic ___
(d) White ___
(e) Other (Please explain: _____)

20. What was the first language that you learned?

21. If your first language was not English, at what age did you learn English?

22. What language do you currently speak at home?

23. Please list any other languages that you currently speak fluently?

Thank you for your help, it is very much appreciated.

Appendix D
Some Examples Presented in Class

1a. (Textbook p. 37)

```
Program InchtoCm;
{Converts number of in to an equivalent number of cm}
Var in: integer;  cm: real;
Begin
     Write ('Enter no. of inches ');
     ReadLn (in);
     Cm := 2.54 * in;
     Write (in, ' inches equals ');
     WriteLn (cm:0:1, ' centimeters');
End.
```

1b. (Class discussion)

(i) Modify program 1a to convert kilometers to miles.

(ii) Modify program 1a to convert feet to yards.

Similarities to original program: Essential structure is unchanged.

Differences from original program: The units of measurement clearly differ; the values of the conversion factors differ; in modification (ii) the conversion is from a smaller unit to a larger unit so the conversion factor is less than 1.

2a. (Textbook p. 102)

```
Program Over1000;
{Finds first square to put sum over 1000}
Var N, Sum: integer;
Begin
     N := 0; Sum := 0;
     While Sum <= 1000 Do
       Begin
          N := N + 1;
          Sum := Sum + N*N;
       End; {While}
WriteLn ('Sum first goes over 100 when you add ', N,
'squared');
WriteLn ('Sum = ', Sum)
End.
```

2b. (Class discussion)

(i) Write a program to find the integer that first puts the sum of the reciprocals of the first N integers over 2.5.
(ii) Write a program to find the greatest integer that will keep the sum of the first N integers under 500.

Similarities to original program: Essential structure of the problem is unchanged.

Differences from original program: (i) The sum is of reciprocals not of squares; the bound is 2.5 not 1000. (ii) The sum is of the integers themselves, not of their squares. You are looking for the greatest integer that will keep the sum under the bound rather than the smallest integer that first puts the sum over the bound.

Appendix E
Programming Worksheets

PROGRAMMING WORKSHEET #1

Directions

1. Carefully read through the base **problem** and the code for the program that implements it.

2. Read through the target problem.

3. Apply the following analogical-reasoning steps to develop a program to implement the target problem.
   (a) Encode: Analyze and make notes about the overall structure of the base problem. Is it sequential? Does it use selection? Does it use repetition?
   Do the same for the target problem. Analyze and make notes about the input and output of both problems, if any.
   (b) Infer: Trace the code in the base program until you understand exactly how it produces the desired results; write a sentence or two to describe how the base program works.
   (c) Map: Write down the similarities of structure between the two problems that you see. Write down the differences of structure between the two problems that you see.
   (d) Apply: Try to use the applicable aspects of the structure of the base program to write the target program.

****

Base Problem

Baseballs are priced at $6.50 each if at least 10 are purchased and at $7.00 each otherwise. Write a program that reads in the number of baseballs and calculates and prints out the total cost.

Code for Base Program

```
Program baseballs;
uses printer;
    Var   num: integer; cost: real;

Begin
  write ('Enter number purchased: ');
      readln (num);

      if  num >= 10 then cost := num * 6.50
                else cost := num * 7.00;
      writeln (lst, num, ' baseballs cost $', cost:0:2)
    End.
```

Target Problem

At a state college, tuition charges are $50 per credit, with a maximum charge of $750 no matter how many credits are taken. Write a program that reads in the number of credits and calculates and prints out the tuition charge.

****

Code for Target Program
(Attach additional pages if necessary.)

****

How did your program work? Did you get the results you were aiming for? How is the structure of the target program analogous to that of the base program? If things aren't quite right, describe in words what you think went wrong:

PROGRAMMING WORKSHEET #2

Directions

1. Carefully read through the base **problem** and the code for the **program** that implements it.

2. Read through the target problem.

3. Apply the analogical-reasoning steps to develop a program to implement the target problem.

(a) Encode: Analyze and make notes about the overall structure of the base problem. Is it sequential? Does it use selection? Does it use repetition? Do the same for the target problem. Analyze and make notes about the input and output of both problems.

(b) Infer: Trace the code in the base program until you understand exactly how it produces the desired results; write a sentence or two to describe how the base program works.

(c) Map: Write down whatever similarities of structure between the two problems that you see. Write down whatever differences of structure between the two problems that you see.

(d) Apply: Try to use the applicable aspects of the structure of the base program to write the target program.

****

(After you have run a first pass of the program, take some time to think about and answer the following questions.)

4. Is the structure of the target program truly analogous to that of the base program? How did the new program work? Did you get the results you were aiming for? If things aren't quite right, describe in words what you think went wrong:

5. Is there a part of the program that you think should be modified?

Base Problem (Adapted from pp.100-101 of textbook.)

The Very Best department store has asked you to write a program to produce customer receipts. The program must read in (a) the name of each item being purchased; (b) the number being purchased; and (c) the price of the item. The END-OF-DATA sentinel is the string 'xyz' typed into the item-name field.

Code for Base Program

```
Program Receipt;
{Program to calculate total receipt of department store
purchases for one customer.}
uses printer, crt;

Var
  Item: string[25];
  Quant: integer;
  Price, ItemCost, ItemTot, TotCost: Real;

Begin
  Clrscr;
  TotCost := 0.0;
  Write ('Enter item name  OR  xyz  to stop: ');
  Readln (Item);

  While (Item <> 'xyz') Do
     Begin
       Write ('  Enter quantity of item purchased and price
per item:  ');
       Readln (Quant, ItemCost);
       ItemTot := Quant * ItemCost;
       TotCost := TotCost + ItemTot;
       Writeln (Lst, Quant, '  ' , Item, ' $', ItemTot:6:2);
       Write ('Enter item name  OR  xyz to stop: ');
       Readln (Item);
     End;   {While}
  Writeln (Lst, 'Total Cost is ', TotCost:7:2);
End.  {Program}
```

Target Problem

The Yum-Yum catering service has asked you to write a program to produce customer receipts. The program must read in (a) the number of adults to be served; (b) the number of children to be served; (c) the cost of an adult's meal; (d) the room fee (there is no room fee if the catering is at the customer's home). You are also told that the cost for a child's meal is 60% of the cost of an adult's meal. The END-OF-DATA sentinel is the value -1 typed into the number-of-adults field.

****

Code for Target Program

(Attach additional pages if necessary.)

PROGRAMMING WORKSHEET #3

Directions

1. Carefully read through the base problem and the Pascal code for the program that implements it.

2. Read through the target problem.

3. Apply the analogical-reasoning steps to develop a program to implement the target problem.

(a) Make notes about the base problem: Clarify what the input is, what the output is, and the processing involved. Do the same for the target problem.

(b) Trace the code in the base program until you understand how it produces the desired results; write a sentence to describe how the base program works.

(c) Write down the similarities you see between the two problems. Write down the differences you see between the two problems.

(d) Make as much use as possible of the applicable aspects of the base problem in writing the Pascal code for the target problem.

(After you have run a first pass of the program, take some time to think about and answer the following questions.)

4. How did the new program work? Did you get the results you were aiming for? If things aren't quite right, describe in words what you think went wrong; How can you change your program to fix its problems?

****

Base Problem

Write a program that (1) Reads in values for four sides of a quadrilateral, S1, S2, S3, S4; (2) Calls a Boolean function IsPgram that determines if the four sides form a parallelogram; (3) If IsPgram returns True, calls an integer function FindPerim that finds the perimeter of the parallelogram; (4) Calls a procedure PrintOut that prints out the values of the four sides and EITHER the message that the four sides do not form a parallelogram OR the value of the perimeter of the parallelogram. (Use the geometrical fact that a quadrilateral is a parallelogram if opposite sides are equal.)

The data for this program should be read from an external file named A:Pgram.dat. All communication between the main program and the functions and procedure **must** be through parameters.

Code for Base Program

```
Program Parlelgm;
uses printer;
VAR
   S1, S2, S3, S4, Per: Integer;
   YesPgram: Boolean; InData: Text;
Function IsPgram (Side1, Side2, Side3, Side4:Integer):
Boolean;
  Begin
    IF (Side1 = Side3) AND (Side2 = Side4) THEN
          IsPgram := True
    ELSE
          IsPgram := False
  End; {IsPgram}

Function FindPerim(Side1, Side2: Integer):Integer;
  Begin
    FindPerim := 2*Side1 + 2*Side2;
  End; {FindPerim}

Procedure PrintOut (Side1, Side2, Side3, Side4, Perim:
Integer);
  Begin
    IF Perim = 0 THEN
       Begin
       Write (Lst, Side1:4, Side2:4, Side3:4, Side4:4);
       Writeln (Lst, ' DO NOT FORM A PARALLELOGRAM')
       end
    ELSE
       Begin
       Write (Lst, Side1:4, Side2:4, Side3:4, Side4:4);
       Writeln (Lst, 'FORM A PARALLELOGRAM WITH PERIMETER ',
                Perim);
       end
  End; {PrintOut}

    Begin {Main}
    Assign (InData, 'A:Pgram.dat');
    Reset (InData);
    While Not Seekeof (InData) Do
    Begin
    Readln (InData, S1, S2, S3, S4);
    YesPgram := IsPgram (S1, S2, S3, S4);
    IF YesPgram THEN
      Per := FindPerim (S1, S2)
    ELSE
      Per := 0;
    Printout (S1, S2, S3, S4, Per);
    End; {While}
    Close (InData)
  END.
```

Target Problem

Write a program that (1) Reads in values for three line segments, a, b, and c, from an external data file named Herodata.dat; (2) Calls a Boolean function IsTriangle to determine if the three sides form a triangle; (3) Calls a real function FindArea that uses Hero's formula to find the area of the triangle if IsTriangle has come back true; (4) Calls a procedure PrintOut that prints out the values of the three line segments and EITHER the message that the line segments do not form a triangle OR the area of the triangle.

Geometrical Information: Three line segments form a triangle if the sum of each two is greater than the third.

Hero's formula for a triangle's area, A, involves the semiperimeter, s:

$$s = (a + b + c) / 2$$

$$A = \sqrt{s\,(s - a)(s - b)\,(s - c)}$$

The data file Herodata.dat should contain the following 5 lines

```
12   16   20
 5   14   23
 5   12   13
 9   40   41
 7   19   27
```

==========

Code for Target Problem

Appendix F
Pretest Reading Passages

Part I--Learning Session

Code: ___ __

(Please enter your code in the upper-right-hand corner of each page: The first three letters of your middle name followed by the day of the month of your birthday. If you have only a middle initial, please enter the initial followed by ON. If you have no middle name or initial, please enter NMI; if the day of your birthday is a single digit, please enter 0 before the digit.)

****

****

After you are told to turn the page, you will be asked to read two stories and to answer some questions about them. Both stories describe problems and the strategies used to solve the problems.

Code: ___ __

Please read story No. 1.

Story No. 1

New members of the military do not usually have the skills needed for battle. They are trained at bootcamp, where they participate in many pretend-war activities. The activities are the same as those encountered in a real war, but they are not life-threatening. For example, if bullets are fired, they are not live ammunition. By practicing these activities the new soldiers gain military battle skills without facing the danger of actual war. Their commanders hope that when they are faced with a battle in a real war all of the practice battles that they have fought will have prepared them with the skills to survive live battle.

Code: ___ __

Think about the following questions carefully before you answer. It's a good idea to refer back to the story as you answer these questions.

Q1. What is the main problem described in Story No. 1?

Q2. What strategy is used to solve the problem?

Code: ___ __

Please read story No. 2.

Story No. 2

Barrow and Morely are the names of planets in a distant solar system. Scientists on planet Barrow have been having problems with the rockets they've been using to explore planet Morely. When the rockets return to Barrow from Morely, most of the outside skin of the rockets has crumbled. The scientists have identified MGAS, a gas in the atmosphere around Morely, as the cause of the crumbling.

Scientific analysis yielded two surprising results: (1) some of the crumbled pieces of the rockets that had returned to Barrow showed that those pieces of the rocket skin that had survived the crumbling had actually been made stronger; and (2) some samples of MGAS brought back to Barrow by one of the rockets showed that MGAS contains a component called Mgas-x that has the property of making the rocket skin stronger.

Because of these surprising results, engineers have begun exposing the rocket-skin material to Mgas-x as part of the rocket-construction process. Their hope is that this will make the rocket skin strong enough to survive its passage through MGAS as it approaches Morely. Thus far, the rockets with skins made out of material treated with Mgas-x have all returned from Morely without showing any crumbling effects.

Code: ___ __

Think about the following questions carefully before you answer. It's a good idea to refer back to the story as you answer these questions.

Q1. What is the main problem described in Story No. 2?

Q2. What strategy is used to solve the problem?

Code: ___ __

It's a good idea to refer back to both stories that you've just read as you answer these questions.

Do you see any similarities between Story 1 and Story 2?

(Think carefully. The problems in the two stories are certainly not the same, but can you see anything about the strategies by which the problems are solved that could be said to resemble each other?)

Appendix F
Part II--Analogy Session

Code: ___ __

(Please enter your code in the upper-right-hand corner of each page: The first three letters of your middle name followed by the day of the month of your birthday. If you have only a middle initial, please enter the initial followed by ON. If you have no middle name or initial, please enter NMI; if the day of your birthday is a single digit, please enter 0 before the digit.)

****

****

The following pages contain some problems to be solved. You will be given up to 7 minutes to work on each problem. Please read each problem carefully. Give a specific solution to each problem. Make sure that your solution does not contradict the information given in the problem.

Code: ___ __

Zerdia is a small land-locked country surrounded by larger countries. Zerdia is known for its artistic and cultural achievements, especially in dance. Zerdia also has a superb educational system, and some of its graduate computer engineers have designed the world's fastest, most sophisticated computers. Zerdia is a weak country militarily so it attempts to maintain good relations with its larger neighbors.

Nevertheless, Zerdia's neighbor Gagrach launched a surprise missile attack against Zerdia. However, the missiles that Gagrach launched had very poor, inaccurate computer-guidance systems, as well as poor detonation systems, and the attack failed. Many of the missiles landed in unpopulated areas and practically none of the warheads detonated. Some of the missiles broke into very large pieces when they landed in Zerdia, so it was possible for Zerdian scientists and engineers to study the missiles.

After considerable study, the Zerdian government hit on a way to negotiate a peace treaty with Gagrach in which Gagrach agreed not to attack Zerdia again.

Code: ___ __

(Take some time to think about the following question before you answer. Do not hesitate to refer back to the previous page.)

Q1. How you think the Zerdian government was able to negotiate such a favorable peace treaty?

Code: ___ __

One summer, the fish in a lake were dying off. Scientists investigated and found that the fish were being killed by drats, tiny organisms that live in water. This was surprising because fish and drats usually lived together in the lake without any problem. When the scientists studied the drats, they found that it was a new variety of drats that had invaded the lake and was poisoning the fish.

One of the scientists thought that it might be possible to use zylots, small frog-like creatures, to kill the drats. A zylot feeds on tiny organisms that live in water and can survive in many different environments on different varieties of food.

The scientist brought a large group of zylots from her lab and put them in the lake, but most of these zylots died. They were not able to digest the new variety of drats that was in the lake that summer. The scientist continued working with zylots in her lab, and some time later she brought a large group of these zylots to the lake. These zylots were able to eat and digest the new variety of drats, and the fish population recovered.

Code: ___ __

(Take some time to think about the following question before you answer. Do not hesitate to refer back to the previous page.)

Q1. How do you think the scientist was able to accomplish her purpose?

Code: ___ __

Three sisters, Alice, Adelaide, and Amelia, grew up in a large city. Although their interests differed (Alice was a dancer, Adelaide a painter, and Amelia a chemist), they shared a longing to spend some time in a remote, rural area. One summer, after all three sisters were married, Alice's husband Arthur suggested that the three couples take a vacation together in a scenic, remote area of Alaska. The sisters were excited about going, but were concerned about how well the vacation would work out because all their husbands were very jealous.

In spite of their worries, the vacation went along quite well until one day they went hiking in an extremely remote area. They were enjoying the beauty of the scenery and came upon a large, beautiful lake with a large island. They found a boat on the lake shore that could be used to take them to the island. Unfortunately the boat could only hold two people at one time, and the husbands were so jealous that none of them would allow his wife to travel over with another sister's husband. (One person had to be in the boat to pilot it at all times.) For a while the group thought that they might not be able to get to the island, but they were able to figure out a way that all six people could get there without any of the women having to ride with another's husband.

Code: ___ __

(Take some time to think about the following question before you answer. Do not hesitate to refer back to the previous page.)

Q1. How do you think all six people were able to get to the island without any of the husbands getting jealous?

Code: ___ __

Appendix G
Posttest Reading Passages

Part I--Learning Session

(Please enter your code in the upper-right-hand corner of each page: The first three letters of your middle name followed by the day of the month of your birthday. If you have only a middle initial, please enter the initial followed by ON. If you have no middle name or initial, please enter NMI; if the day of your birthday is a single digit, please enter 0 before the digit.)

****

****

After you are told to turn the page, you will be asked to read two stories and to answer some questions about them. Both stories describe problems and the strategies used to solve the problems.

Code: ___ __

Please read story No. 1.

Story No. 1

Famed firefighter Red Adair and his crew were working on a blazing oil-well fire after the war in Kuwait. Red knew that the fire could be put out if a huge amount of fire-killing foam could be dumped on the base of the burning oil well. But, even though there was enough foam available, there was no hose large enough. The small hoses that were available could not shoot the foam quickly enough to do much good. It looked as if there would have to be a damaging delay to get a large hose before a serious attempt could be made at fighting the fire.

Red Adair found a solution to the problem. He stationed firefighters in a circle all around the fire. Each firefighter had one of the small hoses. All of the hoses were opened at the same time and foam was directed at the fire from all directions. In this way a huge amount of foam quickly struck the source of the fire. The oil-well blaze was extinguished and additional environmental damage was avoided.

Code: ___ __

Think about the following questions carefully before you answer. It's a good idea to refer back to the story as you answer these questions.

Q1. What is the main problem described in Story No. 1?

Q2. What strategy is used to solve the problem?

Code: ___ __

Please read story No. 2.

Story No. 2

A military government was established in a small country after the elected government was toppled in a coup. The military government imposed martial law and abolished all civil liberties. A tank corps commander and his forces remained loyal to the overthrown democratic government. They hid in a forest waiting for a chance to launch a counterattack.

For the counterattack to succeed the military government's headquarters would have to be captured. The headquarters was located on a heavily guarded island in the center of a lake. Since the tank commander had no boats, helicopters, or airplanes available to him, the only way he could get his troops to the island was over several narrow bridges. Unfortunately, each bridge was so narrow and unstable that only a few tanks, many fewer than would be needed for a successful attack, could cross at one time. Nevertheless, the tank corps commander was able to mount a successful attack. He divided his troops and sent a few tanks over each narrow bridge. All went at the same time. In this way, a large enough number of troops got to the island at once and captured the headquarters.

Code: ___ __

Think about the following questions carefully before you answer. It's a good idea to refer back to the story as you answer these questions.

Q1. What is the main problem described in Story No. 2?

Q2. What strategy is used to solve the problem?

Code: ___ __

It's a good idea to refer back to both stories that you've just read as you answer these questions.

Do you see any similarities between Story 1 and Story 2?

(Think carefully. The problems in the two stories are certainly not the same, but can you see anything about the strategies by which the problems are solved that could be said to resemble each other?)

Code: ___ __

Appendix G

PART II--Analogy Session

(Please enter your code in the upper-right-hand corner of each page: The first three letters of your middle name followed by the day of the month of your birthday. If you have only a middle initial, please enter the initial followed by ON. If you have no middle name or initial, please enter NMI; if the day of your birthday is a single digit, please enter 0 before the digit.)

****

****

****

****

The following pages contain some problems to be solved. You will be given up to 7 minutes to work on each problem. Please read each problem carefully. Give a specific solution to each problem. Make sure that your solution does not contradict the information given in the problem.

Code: ___ __

Satellite Q is a special satellite that was designed to take detailed pictures of planets. It is given electronic signals from scientists at an earth-based control center as to what path to follow around the planet and at what interval to take pictures. When an orbit is completed, Satellite Q sends an electronic signal back to the control center. The scientists at the center then make very careful, detailed calculations as to what orbit Satellite Q should follow next. The process of filming that Satellite Q engages in is so crucial for the detailed surface-feature maps that are to be constructed from the photographs that the control-center scientists make every effort to make sure that there is no overlap between orbits. To verify that a new orbit does not touch on an old one, a checking-process signal is sent to Satellite Q as soon as a new orbit is calculated. This signal activates the verification part of the satellite's picture-taking mechanism very briefly. If the verification is correct, a "verified" signal is sent back to the control center and a new round of picture taking is begun; if not, a "not verified" signal is sent to earth and the orbit is recalculated.

Code: ___ __

(Take some time to think about the following question before you answer. Do not hesitate to refer back to the previous page.)

Q1. Describe briefly what you think is involved in the verification method used by Satellite Q:

Code: ___ __

A radiologist has a patient who has a malignant tumor in his stomach that cannot be operated on. If the tumor is not destroyed, the patient will die. There is a radiation treatment that can destroy the tumor, but if it is used at a high enough intensity, all of the healthy tissue that the rays pass through will also be destroyed. If the rays are used at lower intensities, they don't hurt the healthy tissue, but they aren't strong enough to destroy the tumor. However, in spite of these difficulties, the radiologist found a way to destroy the tumor with the rays without harming the healthy tissue.

Code: ___ __

(Take some time to think about the following question before you answer. Do not hesitate to refer back to the previous page.)

Q1. How do you think the radiologist was able to accomplish her purpose?

Code: ___ __

In the last general election, voters in Orich, a large city, elected both a new mayor and a new city council. It was a very hard-fought election in which the opposing parties had large differences. Therefore, the voters' choices were puzzling. They elected a mayor who greatly favored development; he wanted to see new construction, of both office and residential buildings, in all parts of the city. In contrast, a majority of the city council opposed development; some of the council members even wanted to replace some existing business buildings in the waterfront area with parks. As the next general election approached two years later, commentators remarked on how the city had not increased or decreased much in its level of development.

Code: ___ __

(Take some time to think about the following question before you answer. Do not hesitate to refer back to the previous page.)

Q1. Why do you think that the city of Orich had not increased or decreased in its level of development over the two years described?

Appendix H
Learning-Style Inventory

The Learning-Style Inventory describes the way you learn and how you deal with ideas and day-to-day situations in your Below are 12 sentences with a choice of four endings. Rank the endings for each sentence according to how well you t each one fits with how you would go about learning something. Try to recall some recent situations where you had to l something new, perhaps in your job. Then, using the spaces provided, rank a "4" for the sentence ending that describes how learn *best*, down to a "1" for the sentence ending that seems *least* like the way you would learn. Be sure to rank all the enc for each sentence unit. Please do not make ties.

Example of completed sentence set:				
0. When I learn:	4 I am happy.	1 I am fast.	2 I am logical.	3 I am careful.
1. When I learn:	____ I like to deal with my feelings.	____ I like to watch and listen.	____ I like to think about ideas.	____ I like to be doing things.
2. I learn best when:	____ I trust my hunches and feelings.	____ I listen and watch carefully.	____ I rely on logical thinking	____ I work hard to get things done.
3. When I am learning:	____ I have strong feelings and reactions.	____ I am quiet and reserved.	____ I tend to reason things out.	____ I am responsible about things.
4. I learn by:	____ feeling.	____ watching.	____ thinking.	____ doing.
5. When I learn:	____ I am open to new experiences.	____ I look at all sides of issues.	____ I like to analyze things, break them down into their parts.	____ I like to try things out.
6. When I am learning:	____ I am an intuitive person.	____ I am an observing person.	____ I am a logical person.	____ I am an active person.
7. I learn best from:	____ personal relationships.	____ observation.	____ rational theories.	____ a chance to try out and practice
8. When I learn:	____ I feel personally involved in things.	____ I take my time before acting.	____ I like ideas and theories	____ I like to see results from m work.
9. I learn best when:	____ I rely on my feelings.	____ I rely on my observations.	____ I rely on my ideas.	____ I can try things out for myself.
10. When I am learning:	____ I am an accepting person.	____ I am a reserved person.	____ I am a rational person.	____ I am a responsible person.
11. When I learn:	____ I get involved.	____ I like to observe.	____ I evaluate things.	____ I like to be active.
12. I learn best when:	____ I am receptive and open-minded.	____ I am careful.	____ I analyze ideas.	____ I am practical.

Appendix I
Class Presentation on Analogical Reasoning

(Lecture notes)

Thinking by analogy is quite common.

Whenever you make a decision about something new in your experience by drawing a parallel to something old in your experience, you are thinking by analogy.

For example, if you listen to friend's advice about where to go to college because you think that he gave you good advice about where to go to summer camp, you are making an analogy.

You are implicitly making the parallel "Since his advice was good about the summer camp, he probably will have good advice about college". Whether this is a valid analogy depends on the particular person you are dealing with. In everyday uses of analogy, you often will not know if your analogy is a good one until it plays itself out.

We want to look at some more formal, less everyday uses of analogy, especially analogies that are used in science.

1. A widely used example: Many science textbooks talk about a model of the atom that says that an atom is structured like our solar system.

Let's look at the model.

Similarities

Large sun at center	Large nucleus at center
Large sun attracts small planets	Large nucleus attracts small electrons
Small planets revolve around sun	Small electrons revolve around nucleus

Differences

Sun is very massive	Nucleus is very tiny
Sun is very hot	Nucleus is not very hot
Sun is yellow	Does nucleus have a color?

In formal terms, we refer to the base domain, the solar system, and the target domain, the atom.

What gets mapped? The important things that get mapped in a good analogy are relationships between items that belong to the base domain and items that belong to the target domain. In our example:

The large sun at the center **attracts** the small planets

The large nucleus at the center **attracts** the small electrons

The planets **revolve around** the sun

The electrons **revolve around** the nucleus

What doesn't get mapped? A good analogy does not have to map the attributes of the items in the base domain to the target domain. In our example:

The sun is large, gaseous, hot, and yellow among other things. The planets are composed of many different types of materials and have many different temperatures.

It is not necessary for these attributes of the items in the base domain (sun and planets) to be mapped to the items in the target domain (nucleus and electrons) in order for the analogy to be productive.

2. The analogy between a water pump and a human heart.

Similarities

The water pump pumps a liquid

The human heart pumps a liquid

Differences

The water pump pumps water

The human heart pumps blood

The water pump is made out of metal

The human heart is made out of human tissue

Water is pumped through metal pipes

Blood is pumped through arteries, capillaries, and veins

Appendix J
PROPOSED PROGRAMMING WORKSHEET

Directions

Carefully read through the two problems that follow and the programs written to implement them. Think about the similarities and differences between the two situations presented. Then answer the questions that follow.

Problem No. 1

The Thrify Express delivery company charges a $15 pickup charge at each stop where it makes a pickup. In addition, there is a $10 delivery charge for every package that weighs 2 pounds or less, and a $20 charge for packages that weigh between 2 and 10 pounds. (The company does not handle packages that weigh more than 10 pounds.) Write a program to calculate the total amount collected by a delivery truck during a single driver's shift.

Program for Problem No. 1

```
Program Thrifty;
     Const  pkupchg = 15;
     Var    nostops: integer;
            wt, totdelchg, total: real;
     Begin
     Write ('Enter number of stops made: ');
     Readln (nostops);
     Write ('Enter package weight; enter 0 to stop: ');
     Readln (wt);
     While  wt > 0  Do
       Begin
         If wt > 10 Then
           Writeln ('Package too heavy. Return.')
         Else If wt <= 2 Then
                 totdelchg := totdelchg + 10
              Else
                   totdelchg := totdelchg + 20;
         Write ('Enter package weight; 0 to stop: ');
         Readln (wt)
       End;
     Total := nostops * pkupchg + totdelchg;
     Writeln ('Total collected during shift = ',
                 total:0:2);
     End.
```

Problem No. 2

The Yum-Yum Catering company needs a program to calculate total revenue collected during a week. Company policy is to charge a flat fee of $250 for each event it caters plus $75 per meal for each adult meal that it serves at an event and $35 per meal for each child's meal that it serves at an event. Write the program that Yum-Yum's treasurer can run each Sunday morning to calculate the revenue for the previous week.

Program for Problem No. 2

```
Program Yum-Yum;
     Const  flatfee = 250;
     Var    noevents, mealtype: integer;
            mealchg, total: real;
     Begin
     mealchg := 0;
     Write ('Enter number of events: ');
     Readln (noevents);
     Write ('Enter 1 for adult, 2 for child,
               0 to stop: ');
     Readln (mealtype);
     While  mealtype > 0  Do
       Begin
         If mealtype > 2 Then
           Writeln ('Error. Reenter')
         Else If mealtype = 1 Then
                 mealchg := mealchg + 75
              Else
                 mealchg := mealchg + 35;
         Write('Enter 1 for adult, 2 for child,
                  0 stop:');
         Readln (mealtype)
       End;
     Total := noevents * flatfee + mealchg;
     Writeln ('Total revenue for week = ', total:0:2);
     End.
```

Refer back to the problems and the programs as you answer the questions that follow.

Even though the situations described in the two problems are different, there are certain structural similarities between them. List what you think these structural similarities are:

Trace the code in each program until you are sure how it produces the desired results. Write a sentence or two describing how each of the programs works. What are the input variables in each? the output variables? How are the structural similarities in the problems reflected in the structure of the two programs?

Target Problem

Izzy's Installation Service installs computer cable in office buildings. Company policy is to charge a $100 installation fee for each office visit and $10 per yard for each yard of cable installed. Write a program that Izzy can use to find the monthly total of cash taken in when he does his monthly accounts.

Look back to the two base problems and programs. How is this problem similar to and different from each of the others? Feel free to adapt lines of code from the other programs for use in this new program.

Appendix K
Time Line for Study

I. Year Prior to Commencement of Classroom Instruction

Time prior to instruction:

12 months Location of site and instructors
Review of instructional protocols and measuring instruments

9 months Pilot testing of measuring instruments

6 months Review and analysis of results of pilot testing
Preparation of usable versions of analogical reasoning measuring instruments, programming projects, and class presentations

3 months Purchase of textbooks and Learning Style Inventory

6 weeks Begin two-week training program with course instructors

II. Semester of Classroom Instruction

Week 1 Presentation of project overview to students
Administration of Student Participation Form (Appendix B) and Student Background Questionnaire (Appendix C)

Week 2 Administration of pretest reading passages
Class presentation on analogical reasoning (Appendix I)
Initiation of classroom computer language instruction employing examples based on those in Appendix D

Week 3 Administration of Learning Style Inventory (Appendix H)
Continuation of classroom instruction
Assignment of first programming by analogy worksheet of the semester during last part of last class meeting for week

Week 4 Continuation of classroom instruction
Return of corrected first programming worksheet
In-class exam containing some questions involving analogical reasoning during last class meeting for week

Week 5 Return of in-class exam; class discussion and review of exam including analogical reasoning material
Continuation of classroom instruction

Week 6 Continuation of classroom instruction
Assignment of second programming by analogy worksheet of the semester during last part of last class meeting for week

Week 7 Continuation of classroom instruction
In-class quiz containing some questions involving analogical reasoning during second half of last class meeting for week

Week 8 Return and discussion of quiz
Continuation of classroom instruction

Week 9 Return and discussion of second programming worksheet
Continuation of classroom instruction

Week 10 Continuation of classroom instruction
Assignment of third programming worksheet
Full-period exam containing some questions on analogical reasoning during last class meeting for week

Week 11 Continuation of classroom instruction
Return and discussion of full-period exam

Week 12 Assignment of fourth programming worksheet
Continuation of classroom instruction

Week 13 Continuation of classroom instruction
In-class quiz containing some questions involving analogical reasoning during second half of last class meeting for week

Week 14 Return and discussion of quiz
Return and discussion of fourth programming worksheet
Assignment of fifth programming worksheet, due at Final Exam
Classroom review of semester course material both programming language and analogical reasoning

Week 15 Classroom review of all semester course material
Administration of posttest reading passages

Week 16 Final Exam

Appendix L
Possible Replacement Analogical Reasoning Exercise

The following analogical reasoning model was constructed by Clement and Gentner (1991) to deliberately make use of fictional domains in order to control for the possibility of student subjects' giving responses governed by their prior beliefs about real-world objects rather than by analogical reasoning between the base and target domains.[23]

Base Domain

The base domain describes fictional creatures called tams that live on a planet in a distant solar system.

(1) Tams live on rocks and survive by using their underbellies to grind and consume minerals present in the rock.

(2) Although at birth a tam has a somewhat inefficient underbelly, the underbelly adapts and develops a texture that is specifically suited for grinding the particular rock on which the tam is living.

(3) The specialized underbelly the tam has developed cannot function on a new rock.

(4) If a tam runs out of minerals on the rock where it is living, it must then stop using its underbelly.

Target Domain

The target domain describes robots that exist on planets in another distant solar system and have probes to gather specialized data.

(1) The robots gather data with probes.

(2) The probes are delicate, and the more they are used the more highly specialized they become for collecting a particular type of data.

---

[23]The actual passages presented to students were one page narratives. This appendix presents the salient aspects of the base and target domains in outline form.

(3) If a robot that has been functioning for a considerable length of time runs out of its specialized type of data on a planet, it may have become so specialized that there is no other data that it can collect.

(4) If a long-functioning robot runs out of its specialized type of data on a planet, it has to stop collecting data.

Analogical reasoning exercises based on the domains described above and similar fictional domains will be designed. They will be modeled on those in Appendices F and G and pilot tested and evaluated for use with students in introductory programming language classes.

References

Anderson, J.R. (1983). The architecture of cognition. Cambridge, MA: Harvard.

Anderson, J.R. (1989). The analogical origins of errors in problem solving. In: D. Klahr and K. Kotovsky, eds., 21st Carnegie symposium on cognition.

Association of American Colleges. (1990). Integrity in the college curriculum: A report to the academic community, 2nd. ed. Washington, DC: Association of American Colleges.

Bochenski, I.M. (1970). A history of formal logic, trans. and ed. by Ivo Thomas. New York: Chelsea Publishing Co.

Bronowski, J. (1975). Science and human values, rev. ed. New York: Harper & Row, Harper Colophon Books; first published Julian Messner, 1965.

Clement, C.A. (1994). The effect of structural embedding on analogical transfer: Manifest vs. latent analogs. American Journal of Psychology.

Clement, C.A.; and Gentner, D. (1991). Systematicity as a selection constraint in analogical mapping. Cognitive Science, 15: 89-132.

Clement, C.A.; Kurland, D.; Mawby, R.; and Pea, R.D. (1986). Analogical reasoning and computer programming. Journal of Educational Computing Research, 2(4): 473-485.

Dalbey, J.: and Linn, M.C. (1985). The demands and requirements of computer programming. Journal of Educational Computing Research, 1(3): 253-274.

Dijkstra, E.W. (1982). Craftsman or scientist? In: Selected writings on computing: A personal perspective. New York: Springer-Verlag.

Duncker, K. (1945). On problem solving. Psychological Monographs, 58: (Whole No. 270).

Evans, T.G. (1968). A heuristic program to solve geometric analogy problems. In: M. Minsky, ed., Semantic information processing. Cambridge, MA: MIT.

Falkenhainer, B.; Forbus, K.D.; and Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. Artificial Intelligence, 41: 1-63.

Fay, A.L. (1990). Computer programming instruction: The acquisition and transfer of design skills. Ph.D. diss., University of California, Santa Barbara.

Fay, A.L.; and Mayer, R.E. (1988). Learning Logo: A cognitive analysis. In: R.E. Mayer, ed., Teaching and learning computer programming: Multiple research perspectives. Hillsdale, NJ: Erlbaum.

Fay, A.L.; and Mayer, R.E. (1994). Benefits of teaching design skills before teaching Logo computer programming: Evidence for syntax-independent learning. Journal of Educational Computing Research, 11: 187-210.

Feigenbaum, E.A. (1970) Information processing and memory. In: D.A. Norman, ed., Models of human memory. New York: Academic.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. Cognitive Science, 7: 155-170.

Gentner, D.; and Gentner, D.R. (1983). Flowing waters or teeming crowds: Mental models of electricity. In: D. Gentner and A.L. Stevens, eds., Mental Models. Hillsdale, NJ: Erlbaum.

Gentner, D.; and Landers, R. (1985). Analogical reminding: A good match is hard to find. In: Proceedings of the International Conference on Systems, Man and Cybernetics. New York: IEEE.

Gentner, D.; and Toupin, C. (1986). Systematicity and surface similarity in the development of analogy. Cognitive Science, 10: 277-300.

Gick, M.L.; and Holyoak, K.J. (1980). Analogical Problem Solving. Cognitive Psychology, 12: 306-355.

Gick, M.L.; and Holyoak, K.J. (1983). Schema induction and analogical transfer. Cognitive Psychology, 15: 1-39.

Glaser, R. (1984). Education and thinking: The role of knowledge. American Psychologist, 39: 93-104.

Grandgenett, N. (1989). An investigation of the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning. Ph.D. diss., Iowa State University.

Grandgenett, N.; and Thompson, A. (1991). Effects of guided programming instruction on the transfer of analogical reasoning. Journal of Educational Computing Research, 7(3): 293-308.

Guilford, J.P.; and Fruchter, B. (1973). Fundamental statistics in psychology and education. 5th ed. New York: McGraw-Hill.

Guilford, J.P.; and Hoepfner, R. (1971). The analysis of intelligence. New York: McGraw-Hill.

Halpern, D.F. (1987). Analogies as a critical thinking skill. In: D.E. Berger, K. Pezdek, and W.P. Banks, eds., Applications of cognitive psychology: Problem solving, education and computing. Hillsdale, NJ: Erlbaum.

Halpern, D.F.; Hansen, C.; and Riefer, D. (1990). Analogies as an aid to understanding and memory. Journal of Educational Psychology, 82(2): 298-305.

Hofstadter, D. (1979). Godel, Escher, Bach: an eternal golden braid. New York: Basic.

Holyoak, K.J. (1984) Analogical thinking and human intelligence. In: R.J. Sternberg, ed., Advances in the psychology of human intelligence. Hillsdale, NJ: Erlbaum.

Holyoak, K.J.; Junn, E.N.; and Billman, D.O. (1984). Development of analogical problem-solving skill. Child Development, 55: 2042-2055.

Holyoak, K.J.; and Koh, K. (1987). Surface and structural similarity in analogical transfer. Memory & Cognition, 15(4): 332-340.

Kantowski, E.L. (1974). Processes involved in mathemataical problem solving. Ph.D. diss., University of Georgia.

Kolb, D.A. (1984). Experiential learning: Experience as the source of learning and development. Englewood Cliff, NJ: Prentice-Hall.

Kurland, D.M.; Clement, C.A.; Mawby, R.; and Pea, R.D. (1987). Mapping the cognitive demands of learning to program. In: In: D.N. Perkins, J. Lochhead, and J. Bishop, eds., Thinking: The 2nd international conference. Hillsdale, NJ: Erlbaum.

Linn, M.C. (1985). The cognitive consequences of programming instruction in classrooms. Educational Researcher, 14(5): 14-29.

Mawby, R. (1987). Proficiency conditions for the development of thinking skills through programming. In: D.N. Perkins, J. Lochhead, and J. Bishop, eds., Thinking: The 2nd international conference. Hillsdale, NJ: Erlbaum.

Maier, N. (1931). Reasoning in humans. II. The solution of a problem and its appearance in consciousness. Journal of Comparataive Psychology, 12: 181-194. (Cited in Gick and Holyoak 1983, pp.13ff.)

Mayer, R.E. (1979). A psychology of learning Basic. Communications of the ACM, 22(110): 589-593.

Mayer, R.E. (1981). The psychology of how novices learn computer programming. Computing Surveys, 13(11): 121-141.

Mayer, R.E. (1992). Thinking, problem solving, and cognition, 2nd ed. New York: Freeman.

Mayer, R.E.; Bayman, P.; and Dyck, J.L. (1987). Learning programming languages: Research and applications. In: D.E. Berger, K. Pezdek, and W.P. Banks, eds., Applications of cognitive psychology: Problem solving, education and computing. Hillsdale, NJ: Erlbaum.

Mayer, R.E.: and Fay, A.L. (1987). A chain of cognitive consequences with learning to program in Logo. Journal of Educational Psychology, 79(3): 269-279.

McBer & Co., Training Resources Group. Learning Style Inventory: Self-scoring inventory and interpretation booklet.

McPeck, J.E. (1990). Critical thinking and subject specificity. Educational Researcher, 19(4): 10-12.

Miller, G.A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 63(2): 81-97.

Mortimer, H. (1988). The logic of induction. New York: Halsted (Wiley).

Newell, A. (1980). One final word. In: D.T. Tuma and F. Reif, eds., Problem solving and education: Issues in teaching and research. Hillsdale, N.J.: Erlbaum.

Newell, A., and H.A. Simon (1972). Human problem solving. Engelwood Cliffs, NJ: Prentice-Hall.

Nummedal, S.G. (1987). Developing reasoning skills in college students. In: D.E. Berger, K. Pezdek, and W.P. Banks, eds., Applications of cognitive psychology: Problem solving, education and computing. Hillsdale, NJ: Erlbaum.

Oppenheimer, J.R. (1956). Analogy in science. American Scientist 11: 127-135.

Perkins, D.N.; and Salomon, G. (1987). Transfer and teaching thinking. In: D.N. Perkins, J. Lochhead, and J. Bishop, eds., Thinking: The 2nd international conference. Hillsdale, NJ: Erlbaum.

Perkins, D.N.; and Salomon, G. (1989). Are cognitive skills context bound? Educational Researcher, 18(1): 16-25.

Pirolli, P.L. (1985). Problem solving by analogy and skill acquisition in the domain of programming. Ph.D. diss., Carnegie-Mellon.

Ratterman, M.J.; and Gentner, D. (1985). Analogy and similarity: Determinants of accessibility and inferrential soundness. In: Program of the Ninth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Erlbaum.

Reed, S.K.; Ernst, G.W.; and Banerji, R. (1974). The role of analogy in transfer between similar problem states. Cognitive Psychology, 6: 436-450.

Salomon, G.; and Perkins, D.N. (1987). Transfer of cognitive skills from programming: When and how? Journal of Educational Computing Research, 3(2): 149-169.

Shneiderman, B. (1980). Software psychology: Human factors in computer and information systems. Cambridge, MA: Winthrop.

Shneiderman, B. (1986). Empirical studies of programmers: The territory, paths, and destinations. In: E. Soloway and S. Iyengar, eds., Empirical studies of programmers. Norwood, NJ: Ablex.

Silverman, D. (1993). Interpreting qualitative data: Methods for analysing talk, text, and interaction. Thousand Oaks, CA: Sage.

Simon, H.A. (1979). The information storage system called "Human Memory." In: H.A. Simon, ed., Models of thought. New Haven, CT: Yale.

Simon, H.A. (1980). Problem solving and education. In: D.T. Tuma and F. Reif, eds., Problem solving and education: Issues in teaching and research. Hillsdale, N.J.: Erlbaum.

Solomon, I. (1991). Effects of task context and domain knowledge on analogical transfer of science knowledge. Ph.D. diss., City University of New York.

Sternberg, R.J. (1977). Intelligence, information processing, and analogical reasoning: The componential analysis of human abilities. Hillsdale, NJ: Erlbaum.

Turkle, S. (1980). Computer as Rorschach. Society, 17(2): 15-24.

Turkle. S.; and Papert, S. (1990). Epistemological pluralism: Styles and voices within the computer culture. Signs, 16(11): 128-156.

Wesselkamper, M.C. (1983). Developing cultural awareness and sensitivity: An experiential approach. D.S.W. diss., City University of New York.

Winston, P.H. (1980). Learning and reasoning by analogy. Communications of the ACM, 23(12): 689-703.

Wirth, N. (1971). Program development by stepwise refinement. Communications of the ACM, 14(4): 221-227.