DOCUMENT RESUME

ED 341 366 IR 015 361

AUTHOR Wideman, Herbert H.; Owston, Ronald D.

TITLE Expert System Development in the Classroom: Processes and Outcomes. Technical Report 91-1.

INSTITUTION York Univ., North York (Ontario). Centre for the Study of Computers in Education.

SPONS AGENCY Social Sciences and Humanities Research Council of Canada, Ottawa (Ontario).

PUB DATE Mar 91

CONTRACT 410-89-1575

NOTE 55p.

PUB TYPE Reports - Research/Technical (143)

EDRS PRICE MF01/PC03 Plus Postage.

DESCRIPTORS Abstract Reasoning; Analysis of Covariance; *Cognitive Processes; Comparative Analysis; Computer Assisted Instruction; Computer Software; *Expert Systems; Foreign Countries; Grade 8; *Intermode Differences; Junior High Schools; Microcomputers; *Problem Solving; Weather

IDENTIFIERS Ontario

ABSTRACT

This study examined cognitive processes and outcomes associated with student knowledge base development. Sixty-nine students from two grade 8 classes were randomly assigned to one of three groups: a knowledge base development (KBD) group, a problem-solving software group, and a control group. Those in the KBD group received relevant instruction and then worked in small teams to develop very simple expert systems for weather prediction for about 20 hours. Students in the software group engaged in problem-solving activities using The Factory and Super Factory; control groups students completed weather instrument projects. Multivariate ANCOVA results for several measures of cognitive skill gain and transfer found no differences between groups. But for those students who scored higher than the grand median on a standardized pretest of abstract reasoning there were significant main effects favoring the KBD group on a formal reasoning test and a transfer task. Based on both the quantitative and the observational data, it was concluded that expert system creation can be a viable means of promoting cognitive development for more advanced students. A tree diagram for a sports knowledge base (KB), a copy of the weather project assignment for developing an expert system, and a student knowledge base chart are appended. (43 references) (Author/DB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED341366

Expert System Development in the Classroom: Processes and Outcomes

Herbert H. Wideman
Ronald D. Owston

Technical Report 91-1
March, 1991

Centre for the Study of Computers in Education
York University

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

H.H. Wideman

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

015361

BEST COPY AVAILABLE

ERIC

Expert System Development in the Classroom: Processes and Outcomes

Herbert H. Wideman
Ronald D. Owston

York University
Centre for the Study of Computers in Education
North York, Ontario

Technical Report 91-1
March, 1991

The research described herein was supported by Social Sciences and Humanities Research Council of Canada Research Grant No. 410-89-1575.

Abstract

This study examined cognitive processes and outcomes associated with student knowledge base development. Sixty-nine students from two grade eight classes were randomly assigned to one of three groups: a knowledge base development (KBD) group, a problem-solving software group, and a control group. Those in the KBD group received relevant instruction and then worked in small teams to develop very simple expert systems for weather prediction for about twenty hours. Students in the software group engaged in problem-solving activities using The Factory and Super Factory; control groups students completed weather instrument projects. Multivariate ANCOVA results for several measures of cognitive skill gain and transfer found no differences between groups. But for those students who scored higher than the grand median on a standardized pretest of abstract reasoning there were significant main effects favouring the KBD group on a formal reasoning test and a transfer task. Based on both the quantitative and observational data, it was concluded that expert system creation can be a viable means of promoting cognitive development for more advanced students.

Table of Contents

Introduction and background

Recent studies forcefully suggest that students can complete 12 or 13 years of public schooling without developing much competence as a thinker (Nickerson, 1988/1989). A relative lack of academic achievement has also been widely reported (Brainin, 1987; Bransford et al., 1986). Many students demonstrate only a superficial understanding of the concepts, relationships, and procedural strategies that are fundamental to the subjects that they have studied. While we are far from understanding all of the elements that have contributed to this state of affairs, recent cognitive research points to several contributing factors. There is a general consensus that commonly employed rote teaching practices are very unlikely to foster thinking, and that they encourage the development of shallow knowledge structures which impede the students' growth of understanding and which are typical of the domain novice (Nickerson, 1988/1989). In addition, in the standard classroom there is usually little explicit teaching of cognitive strategies in ways that can foster skill transference. Perhaps more importantly, students are given little assistance in acquiring and applying the general heuristics and metacognitive skills that they require if they are to assume active responsibility for their own learning and problem-solving; teachers tend to perform these tasks for the student, and normally do nothing to encourage a transfer of the responsibility (Scardamalia & Bereiter, 1989). The results of training studies indicate that successful generalization and transfer of learned strategies only occurs when students are guided to assume the self-monitoring and strategy management functions typically performed by the teacher (Meichenbaum, 1986). Finally, there is the critical but often overlooked element of motivation. The fragmented and apparently useless information and skills which students are supposed to master are often lacking in enough personal meaning to lead students to make the cognitive efforts necessary to learn and develop (diSessa, 1988; Pea, 1988).

Several forms of computer-based and computer-supported learning environments have been devised that are intended to provide both context and support for developing a deeper understanding of the domain under study while at the same time fostering the development of generalizable cognitive and metacognitive learning and problem-solving abilities (for example, Logo (Papert, 1980) and CSILE, the computer supported learning environment (Scardamalia et al., 1989)). Of these, the Logo microworld has been the subject of most study (Clements, 1985). Research results have been mixed. Those investigations reporting positive transfer of

cognitive and metacognitive competencies have made use of enriched, highly mediated learning environments (Littlefield et al., 1989). Unfortunately Logo's application across the curriculum is made problematic by its primary focus on geometric concepts.

Another computer-based environment that appears to offer considerable educational potential is the *expert system shell*, which students can use as a tool for investigating certain types of knowledge domains.

What are expert systems?

Expert systems may be defined as computer-based systems that can replicate human reasoning in a specific area. They are designed to solve problems or make decisions that would normally require the skills of a person highly competent in the problem domain. Typically the expert system queries the user, drawing out the information needed to define the problem and to formulate a solution. The term *expert system* is in some ways a misnomer since these systems are often used for making decisions in areas that do not necessarily require a high degree of professional expertise for competent performance. The domain need only be able to be stated in explicit decision rules, such as those a loan clerk might use for grading a bank loan applicant's risk level.

Expert systems have two key components: a knowledge base and an inference engine. The knowledge base (KB) consists of the facts and concepts of the domain, as well as the rules which define the relationships between these facts and possible outcomes (the range of goals or decisions)[1]. The inference engine uses the questions, facts, and rules in the KB to reach a solution. It contains routines for selecting and applying the rules relevant to solving the problem. The rules themselves are typically expressed in the form IF *antecedent* THEN *consequent*, such that if the antecedent is found to be true (either through querying the user or through logical inference from other rules and facts) the consequent is also held to be true. Rule clauses can be linked by logical (Boolean) operators, as in the example "IF a rectangle has sides of equal length *and* one interior angle is a right angle THEN the rectangle is a square". Rules of this form are known as production rules.

[1]Expert systems sometimes use another structure known as the frame to define these relationships; however, the present discussion will be restricted to rule-based systems.

Expert system shells greatly simplify the development of expert systems by providing a generic inference engine into which a KB can be inserted, eliminating the need for any programming on the part of the developer. The shells also provide development tools for the entry and editing of facts and rules and for evaluating the completeness and logical integrity of the KB. Within the last five years, a number of simple and inexpensive shells that can be used on personal computers have have become available, making their use in the classroom possible.

Expert system development as a pedagogical activity: A rationale

When the critical elements of the KB construction[2] process are considered in the light of recent research and theoretical work on cognition and cognitive development, its potential as an educational activity becomes apparent. There are several grounds for hypothesizing that novice knowledge engineering may foster higher-order reasoning and metacognitive skills while at the same time deepening students' understanding of the subject domains addressed in the knowledge base. First, rule development during KB construction requires the use of a type of inferential thinking essential to the attainment of cognitive competence in a broad range of domains. Some cognitive theorists (e.g. Anderson, 1983; Greeno & Simon, 1988) consider production systems and rules to be the fundamental way in which problem-solving procedures are cognitively encoded. In his widely cited ACT* theory of thinking and learning, Anderson (1983) asserts that mental activity basically consists of production rules being activated. Greeno and Simon (1988) contend that

> A consensus has developed that human knowledge underlying cognitive action can be represented in the form of production rules....In a production system, the basic problem of choice among actions is solved by specifying conditions that lead to the selection of each action that can be performed. The condition of each production rule is a pattern of information that the system can recognize. These patterns include features of the external problem situation (the stimulus). They also include information that is generated internally by the problem solver and held in short-term memory. The internal information includes goals that are set during problem solving. It also can include information in memory, such as past attempts to achieve specific goals. Thus, production rules, which represent basic action knowledge, consist of associations between patterns of information and actions. An action is chosen when the individual has a goal with which the action is associated,

[2]Technically speaking, a KB can be constructed without it being entered into an expert system inference engine, but for the sake of simplicity, and in order to avoid repetitiveness, the terms *KB construction* and *expert system development* will be considered synonymous. *Knowledge engineering* is another expression for KB development that will also be used here.

> and the external stimulus situation as well as information in memory include features associated with the action. (p. 590)

Rules, then, embody *procedural* knowledge—knowledge for carrying out mental activities (*how* to accomplish an end). The other main type of knowledge—*declarative* knowledge—consists of learned facts and propositions (knowing *what*). It is located in the memory network.

Cognitive production rules may be domain-specific, or they may embody more general problem-solving heuristics (sometimes referred to as *weak methods*) which are used to attack problems when the precise way to proceed is not known. Research has uncovered several weak methods that are widely used, including means-ends analysis, working forward, and working backward (see Lesgold, 1988 for a discussion).

It seems plausible, given what we know about fostering cognitive skills, to hypothesize that the explicit, externalized use of production rules during the process of knowledge engineering may lead to a greater awareness of the utility and broad applicability of this form of procedural thinking, which in turn may foster its transfer to other contexts. Evidence from a number of studies indicates that the process of students' making explicit cognitive strategies usually employed implicitly (if at all) can promote strategy mastery and transfer and enhance metacognitive functioning (Brown, Bransford, Ferrara, & Campione, 1983). And it has been repeatedly shown that the likelihood of effective transfer can be enhanced by providing practice in its application in several different contexts (Derry & Murphy, 1986; Shuell, 1986). Because KB construction can be undertaken in a wide range of domains, it provides the opportunity for such practice. As well, students need to be aware of the practical utility of the thinking skills they are being called upon to use for transfer to be likely (Collins, Brown, & Newman, 1988). The broad applicability of production rule reasoning makes it easy to devise knowledge engineering projects that have a high level of salience and intrinsic interest for students, increasing the chances for skill generalization.

KB development is a learning activity that can serve to integrate the use of general problem-solving strategies such as inferential reasoning and means-ends analysis with the acquisition of domain concepts and skills, reestablishing the natural relationship between problem-solving and domain knowledge that has been lost in much current educational practice, a loss which has contributed to the lack of meaning many students experience in schools (Glaser, 1984). Vital

synthesis skills, which can only be developed through constructive activities such as those required for KB engineering, and which are typically ignored in standard curricula, are given an opportunity to grow through their explicit application and practice within students' domains of study (Soloway, 1988).

The process of KB construction should also promote metacognitive awareness and self-regulation. During the process of mapping out and integrating the domain's production rules, student deficiencies and misconceptions in rule representation and procedural knowledge of the domain will get "built into" their expert systems. When tested, these systems will either fail to run or will present incorrect or incomplete solutions to presented problems. The failures and inadequacies of the students' conceptual models will stand revealed, forcing students to seek out and repair incorrect or incomplete model elements to a degree that far surpasses the level of similar activity typical of rote, text-based learning. To be successful, students must iteratively monitor and evaluate their models and debug their production rules, selecting and activating strategies to develop more correct representations. Repeated practice in these metacognitive skills of self monitoring and strategy selection has been shown to be essential to the development of problem-solving facility in several other contexts (Nickerson, 1988/1989). And by compelling systematic elaboration of the linkages and procedural relationships within a domain, KB development should help students form a deeper and more integrated understanding of their subject matter. Studies of novice-expert differences have found that while novices typically pay attention to the surface features of a domain, missing much of its underlying relational structure, experts develop deeper and more cohesive representational schemata that map out the domain's critical concepts and internal relationships (Bereiter & Scardamalia, . Butterfield & Nelson, 1989; Greeno & Simon, 198 The growth of expertise in a domain is characterised by an increase in the procedural utilization of what is initially declarative knowledge (Dreyfus & Dreyfus, 1986). The more complex and proceduralized schemata of the expert make it possible for him or her to more accurately classify problem types and select appropriate solution strategies from their large repertoire of production rules. Knowledge engineering requires such proceduralization and schemata elaboration, and so may promote the development of expertise with a consequent increase in domain problem-solving skill.

Preliminary studies of student knowledge engineering offer some support for this line of argument. In a case study of the development of knowledge bases on projectile motion by six

freshman college physics students, analysis of "thinkaloud" protocols recorded during the 16 to 23 hours of development time indicated that students were able (to varying degrees) to elaborate, refine and proceduralize their declarative knowledge of the domain (Lippert, 1988). Those more successful at the task showed an improvement in their domain problem-solving ability as measured by a posttest. Students reported that the exercise heightened their capacity for managing and monitoring their cognitive resources and evaluating outcomes.

Trollip and Lippert (1987) had graduate students in education work in small groups to create expert systems designed to assist CAI courseware developers lay out their screen displays. The knowledge bases were developed from information garnered in interviews with experts and a literature review. Participants found the exercise highly motivating, and found that much restructuring and refinement of knowledge occurred during the process of implementing their knowledge base in an expert system shell. They considered it a valuable and viable instructional tool.

Much less is known about the value of novice knowledge engineering as an educational activity for younger students. Wideman and Owston (1988) conducted a qualitative observational study of the creation of knowledge bases in a grade seven classroom. After receiving training in the process of knowledge engineering and the use of a rule-based expert system shell, students worked in groups of three to develop expert systems for the taxonomic classification of living organisms. Each group was assigned a project based in one of seven broad categories of organisms, for example, birds or fish. Groups were given a list of a number of varieties of organisms within their category and told that their expert systems must be able to (1) identify one species for each variety, and (2), classify the species according to its grouping at the class and order taxonomic levels. Texts and library materials were used as sources of information. Prior to implementing their KB in the expert system shell, groups were required to map it out by creating a decision tree of the classification process. The teacher monitored group work and provided coaching as necessary. All but one of the seven groups was able to successfully create a working expert system over a period of approximately thirty hours. Their systems contained between twelve and twenty rules, many of which were lengthy, containing four or more clauses. Observations indicated that most students found the work interesting and were able to work effectively in their groups most of the time. Certain deficiencies in domain representation and proceduralization of knowledge would typically emerge at one or more points as the groups' efforts progressed. For example, students would often be unable without some coaching to

conceptually integrate the various levels of classification required in a manner that made possible the proper articulation of the KB in their tree diagrams. Branches might be incomplete or sections skipped. Occasionally the relationship between a classification level (say, *genus*) and the proper *value* for a particular branch of the decision tree at that level would not be grasped, leading to problems in rule development. In general, the deficiencies in cognition encountered were of a type common to novices in a variety of content domains. The difficulties were nearly always surmounted, usually as a result of modelling and coaching provided by the teacher or researchers. Students showed clear evidence of gains in their ability to develop complex domain models by the end of the project, and the researchers concluded that expert wsystem development could be a viable and valuable instructional process.

While the findings from these initial studies are encouraging, we still know little about whether knowledge engineering projects develop any cognitive skills that transfer into other contexts. Given the time and effort required to successfully undertake these projects in the classroom, much more needs to be known about whether they aid students in gaining a deeper mastery of their project domain, and the types of cognitive development they might foster. We have seen how expert system work may promote procedural thinking, foster the integration of domain knowledge, and enhance aspects of metacognitive functioning. The present study was designed to assess whether knowledge base development activities can bring these potential benefits to grade eight students.

The study

Our research was designed to extend earlier investigations of knowledge engineering in two fundamental ways. It provides a quantitative assessment of students' cognitive skill gain in both near and far transfer contexts. Second, it incorporates a true experimental comparison across three different instructional conditions. In the first condition, students carried out expert system development work; in the second, students undertook problem-solving activities in another computer environment; and in a third, students engaged in more traditional classroom work. This design allowed for some initial assessment of the relative value of two different computer-based approaches to promoting higher-order thinking.

Method

Subjects

The subjects were 69 grade eight students from a middle class grade 4-8 school at the fringes of a major metropolitan area. Students at the school had had very little prior exposure to computers in school prior to the start of the study. They were randomly assigned to one of three groups: a KB development group, a Factory activity group, and a control group which undertook a more traditional non-computer-based science project.

Measures

Pretests

All students were administered three pretests from Form A of the Canadian edition of the Differential Aptitude Tests (DAT) battery—Verbal Reasoning, Abstract Reasoning, and Space Relations (Bennett, Seashore, & Westman, 1988). The reported split-half reliabilities ranged from .87 to .95 at the grade eight level. Reported alternate-form reliability at the grade eight level is .94. The battery has been favourably reviewed and is widely used for assessing a range of cognitive abilities in subjects from grade eight through college. It was considered to be the best available published battery for assessing broadly defined cognitive abilities that might be affected by the experimental treatments.

The DAT Verbal Reasoning test is designed to assess the ability to understand and manipulate ideas framed in words at an abstract level. The items "sample the students' knowledge and his or her ability to abstract and generalize relationships inherent in that knowledge" (Bennett, Seashore, & Westman, 1988, p. 7). The Abstract Reasoning Test is a nonverbal measure of students' reasoning ability that assesses the capacity to perceive relationships and generalize principles from nonlanguage designs. Finally, the Space Relations Test is a measure of ability to visualize and structure an object from a pattern or plan. All three tests were group administered following the published procedures.

Posttests

Form B of the same three DAT tests used for pretesting were administered as posttests. Several additional instruments were also used: the Arlin Test of Formal Reasoning (Arlin, 1984); the Student Thinking about Problem Solving Scale (STAPSS) (Armour-Thomas & Haynes, 1988); and four tasks taken from the Thinking Skills Language Program series (1978).

The Arlin test is a standardized and normed paper and pencil measure of an individual's ability to use the eight specific concepts associated with the stage of formal operations as specified in the Piagetian theory of development. By name, these are: (1) multiplicative compensations; (2) correlations; (3) probability; (4) combinations; (5) propositions; (6) forms of conservation beyond direct conservation; (7) mechanical equilibrium; and (8) the coordination of two or more systems of reference. The test manual reports Hoyt estimates of reliability ranging from .71 to .89 depending upon the age of the subjects being tested and the particular version of the test used. A multitrait-multimethod validity study conducted by the test's author concluded that the test is a valid and reliable measure of formal operations. Several favourable reviews of the test have been published (e.g. Mitchell, 1985).

The STAPSS is a group administered self-report questionnaire "designed to assess students' awareness and use of higher-order cognitive processes during problem solving" (Armour-Thomas & Haynes, 1988, p. 88). Each item describes an approach to problem-solving and asks students to indicate the extent to which each statement describes their problem solving approach. Items were generated from the higher-order processes Sternberg has identified in his Triarchic theory of intelligence as being particularly important in problem solving. These include: (1) defining the nature of the problem; (2) selecting the steps needed to solve the problem; (3) selecting a strategy for ordering the steps; (4) selecting a mental representation for the information in the problem; (5) allocating resources to problem solving; and (6) monitoring solution (Sternberg, 1984). Six interpretable orthogonal factors, labelled Organizing, Accommodating, Evaluating, Strategizing, and Recapitulating, were identified in a large-scale administration of the test. Cronbach's alpha, a measure of scale reliability, was found to be .75.

The Thinking Skills Language Program series (1978) is a domain-independent general thinking skills program providing training and practice in a number of reasoning skills over a wide

ability range. Four problem tasks from the series were selected to serve as measures for assessing the transfer of certain problem-solving skills employed in the KB development process. The tasks differed in their transfer distance from the expert system activities. This distance was gaged using criteria outlined in Butterfield and Nelson's (1989) recent restatement of the Common Elements Theory of transfer. The theory holds that

> Transfer will be strong and positive to the extent that a student has encoded previously all and only critical elements [representations, strategies, knowledge] in the relational structure of a learned mental model and has selected all and only critical elements in the representation of the current problem. Transfer will be weakened to the extent that a student selects noncritical elements as if they were critical. (Butterfield & Nelson, 1989, p. 16)

From this perspective, near transfer tasks are defined as tasks that vary only in their noncritical elements; the sequence of critical elements required for completion (the representations, strategies, and knowledge) remains identical. Butterfield and Nelson cite a balance beam experiment by Day and Hall (1988) as an example of a near transfer test. Following subjects' initial training in the balance problems, the number of pegs on each side of the fulcrum, a noncritical element, was raised from four to ten. In the same study, a far transfer task was also employed, in which new critical elements were introduced into the problem:

> Instead of a stack of weights on a peg on each side of the fulcrum, weights were hung in a basket from a peg on each side. Solving the far transfer task required use of a strategy not required by learning, maintenance, or near transfer tasks. (p. 20)

The first transfer task problem used in the present study (task 1) required the hierarchical classification of actors and their actions using a tree chart. Students read a short story involving several characters. They were then required to determine the criterion for a dichotomous classification of the characters and to place the criterion values in the right boxes on a preprinted tree chart. The names of the characters then had to be entered into appropriate subordinate boxes under the right criterion value, and a list of various descriptors of the characters and their actions (e.g., "Hired gunman") had to be correctly classified by character and written on the chart under the appropriate character's name. Students were scored on the basis of the number of correct entries they were able to make into the chart. This task was a far transfer measure, since it required a change in one critical representation—from the predictive classification structure required in the expert system assignment to a branched categorical classification hierarchy.

In task two, students read a factual passage about blood and its components. Based on the information in the passage, they were required to complete a hierarchical tree diagram of four blood components and their functions. They worked with a preprinted tree chart in which a few of the boxes were already filled in to provide guidance about the form to be followed. Students were again scored on the number of correct chart entries they made. This was another far transfer measure since it required the development of a functional rather than a predictive hierarchical mapping.

The third task required students to restate factual information presented in a written passage into production rule form, although the rule syntax required was different than that used in the expert system shell. Students read a short paragraph which explained the economics law of "supply and demand". They were then required to fill in the terms for three mock equations using phrases provided that were derived from the paragraph, such as "price rises" and "high supply". One of the required equations, for example, was "low supply + high demand = price rises". Students were graded on the total number of terms correctly entered into the blank equations. Since the task assessed the ability to translate information presented in text into predictive production rules incorporating clauses linked by boolean operators, it was identical in its critical components to the expert system task and so can be considered a near transfer measure. While there were differences in the syntactical representations required for the rules, these differences were minor.

In the final task, the problem given was to convert information in a map of the West Indies into a four level hierarchical tree diagram. Students were given a classification tree of the solar system as a model for mapping structure. The hierarchical levels of the region and country names on the map could be determined by examining the relative size of the print. Ratings were based on the number of errors made in completing the tree. Errors that were an inevitable consequence of other errors were not counted, since one error (such as placing a region at the wrong hierarchical level) could displace subordinate classifications which might in all other respects be correct. This was considered to be a very far transfer task, since it required the use of a totally new strategy—the conversion of map rather than textual information into a hierarchical tree—as well as the use of a different type of knowledge (classificatory rather than predictive).

Design

The study was undertaken over a three month period in the winter and spring terms of the school year. Following two periods of group pretesting, all students were given three 80 minute sessions of instruction in key meteorological terms and concepts by the science teacher. The teaching methods used were direct whole-group instruction, film viewing, and student worksheet exercises. The three groups then began meeting simultaneously in separate classrooms every second day for the first 80 minutes of the morning. Twelve meetings were held by each group, resulting in a total of 16 hours of class time being devoted to the project. The expert systems (ES) group worked in a classroom containing 10 IBM PC computers equipped with hard drives that were stationed around the periphery of the room. The *Personal Consultant Easy* expert system shell software from Texas Instruments (1986) was loaded on each machine's hard drive. The Factory activity group met in a double classroom; one half of the room had desks in rows facing the front blackboard, and the second half had ten IBM PC jr computers placed on desks around the periphery of the room. This group made use of two related problem-solving environments: *The Factory* and *Super Factory*, software published by Sunburst (n.d.). Students in the project group met in the workshop classroom. Group posttesting was conducted with all students in the week following the end of the class sessions.

All group work sessions were closely observed throughout the course of the study; the expert system group primarily by the investigators (see below), the other two groups by trained research assistants whose observation and notetaking was monitored by the primary researchers. The collection of detailed and accurate descriptions of student activities and work processes was considered essential to the development of a qualitative understanding of the strategies students employed and the strengths and deficiencies of these strategies.

Materials

The expert system shell

The *Personal Consultant Easy* expert system shell is a rule-based shell that runs on IBM PC's and compatibles. It has environments for consulting existing knowledge bases as well as developing new ones. A description of the steps involved in creating a simple expert system with this shell follows in order to make clear the types of strategies and knowledge representations students needed to employ in order to build working systems.

The first step involves defining the domain and the problem that you wish to solve in the relevant domain. For example one might decide to limit the domain to the national park system, and to set the problem as the selection of a park for a holiday. Then the information required for solving the problem must be gathered—in this case, determining the criteria that people typically use for selecting a park site and then amassing knowledge about how the various parks in the country are rated on the various criteria. The criteria might include climate, available sports, location, and lodging or camping facilities. This domain knowledge must then be transformed into a form that can be entered into the shell. PC Easy makes use of two basic knowledge structures that must be entered: parameters and rules. Parameters are conceptual knowledge elements that take one or more values. For example, in the parks case, one parameter might be *location* and its value for one park might be *southern Ontario*. The rules express relationships among the parameters and conclusions about them. For example if the parks expert system determined through questioning a client that he wished to go to a park in southern Ontario that had trout fishing, the following rules might be appropriate and be activated or "fired" to reach a choice:

If LOCATION is southern Ontario and SPORT PREFERENCE is trout fishing*then* PARK is Pelee.

Before entering the parameters and rules to be used, the goal must be defined. In PC Easy the goal is a special type of parameter whose value is determined by the inference engine in the expert system consultation. That value is the conclusion that the expert system reaches. The search for the goal's value is the force that drives the consultation. When one chooses to create

a new knowledge base from the opening PC Easy menu, the shell requests the name of the goal immediately. It then requires that you select the type of goal. In the case of the students' projects, all the goals were singlevalued; only one value of the goal was correct for any one consultation.

Once the goal is set, the domain information must be studied and the factors needed to solve the problem must be selected out and set as parameters for the shell. Several parameters for the parks knowledge base have already been suggested. Their specific values for the various parks have to be determined as these would be used in rules. Parameters and their associated prompts are then entered into the shell by selecting "parameters" from the knowledge base menu after entering the goal. A parameter list appears; at this stage it has only one entry, the name of the goal that had been entered earlier. The Alt-A key combination is hit to bring up the field for entering a new parameter name. Once it is named, its *prompt* is added. This is the question that is asked of the client by the expert system during a consultation to determine the appropriate value of the parameter. For the parks knowledge base parameter location, for example, the prompt might be "Select the location from the list below that you would prefer to visit". Then the type of parameter is selected; *singlevalued* if only one choice is acceptable, *multivalued* if more than one can be chosen. Then an *expect* type is selected. This value tells PC Easy what to expect a client to input for this parameter (a number, or a multiple choice selection, for example). In all student projects, *user-defined* was chosen here so that the author of the knowledge base could then enter a list of possible parameter value choices which the client had to choose from. In entering the values, quotations had to be placed around the separate choices for syntactical reasons.

Once the parameters had all been entered in this manner, the domain rules extracted from the source material could be added to the knowledge base. PC Easy formulates rules using an IF...THEN syntax. New rules are added from the rules list screen, again using the Alt-A key combination. The shell first prompts for entry of the IF clause of the rule. The following syntax is required:

PARAMETER1 = "VALUE A"*and/or/not* PARAMETER2 = "VALUE....

As can be seen, rule clauses were linked using Boolean operators. Parentheses could be used to change the processing sequence.

Once the IF clause was entered the user would be prompted for the THEN clause . Here the goal parameter would be stated with the appropriate value for the rule's IF clause would be entered, e.g.

PARK = "Pelee".

The rest of the knowledge base rules would then be entered the same way.

During the entry of the rule clauses, the shell provided real-time checking of some elements of the syntax. If an error was detected, it would present the clause again for editing. However certain problems, such as differences between the spelling of a parameter name when it was initially defined and when it was referred to in a rule, could not be detected until the system was consulted. The shell would then either deliver a very general error message that would not pinpoint the problem or simply fail to find a problem solution. Students would have to go through all of their rule and parameter screens looking for discrepancies.

The Factory and Super Factory

The Factory provides a simulated factory assembly line on which learners select "machines" to create geometric products. There are three machines. Each performs a different "operation" on the product. One "machine" punches from one to three circles or squares, another machine lays down horizontal stripes that are either thin, medium, or thick, and a third rotates products from 45-180 degrees. The type and number of machines used, and the order in which they operate, results in products of varying complexity. When working in the program, students can test the machines to see their effects, build factories to create objects they have previously drawn and then test them out to see if they produce the identical object, or take one of the computer "challenges" in which the software presents an object to be created by the learner using the factory machines. Challenges can be at one of four difficulty levels, ranging from simple one or two machine designs to complex figures requiring eight operations in a certain sequence. The stated goal of the software is to foster visual discrimination and the the development of spatial and abstract reasoning. It is the software authors' contention that the preparation of the products and the analysis of how they can be made provides a wide range of challenging tasks for students at the junior and intermediate levels.

The optional teacher's workbook (Blake, 1986) provides a number of curriculum activities designed to help students master the steps involved in building products with The Factory. Some are challenges to be tried using the software; there is also a progressive series of tasks in which students create and use factory machine models to experiment with and learn about the various factory operations and their effects. Many of these workbook exercises were utilized in the Factory activity class.

Super Factory extends the concept of the original Factory program by adding a third dimension to the products that are to be produced. Instead of creating patterned squares, the Super Factory manufactures cubes, but the types of patterns available to be placed on the cube's faces are identical to those found in The Factory. And like The Factory, it offers varying difficulty levels; at its most complex it requires several rotations along different axes in conjunction with pattern stamping to produce the "challenge" objects.

Procedure

Expert Systems group

Group instruction in knowledge engineering and the use of the expert system shell was provided by the researchers. In the first session, the purpose of expert systems was discussed. Students were then assigned to groups of three and began consulting a demonstration knowledge base provided with the shell. The knowledge base offered advice about which national parks might be appropriate holiday destinations given the user's recreational interests, geographical location, and so on. Students were shown how specific answers to the questions asked by the expert system led to certain park choices, and the nature of branching as used in the knowledge base was discussed. The class worked together to list characteristics of an expert, using the examples of a doctor and a weather forecaster. The embedding of expertise in rules was illustrated with examples from the parks knowledge base. The formulation of knowledge in the form of production rules was modelled with several examples, and students were asked to provide rule formulations from their work with the parks expert system.

The second session began with a short overview of the utility of expert systems in a number of areas. Their increasing use in medicine, science and business was discussed. Questioning was used to prompt student review of the structure of the rules used in expert systems, with care

being taken to distinguish between a parameter and its value in a given rule. The researchers then modelled the development of a very simple three rule knowledge base that could categorize a geometric form as a circle, triangle, or quadrilateral. Students were asked to supply the defining characteristics and to embed them in rules. Only a few students in the class were able to offer useful suggestions at this point. The restatement of the rules in production format was modelled and the creation of a decision tree flowchart for mapping the parameters and rules illustrated on transparencies. A computer screen projection device was then used to show students how the parameters, their values, the related user questions, and the decision rules were to be entered into the expert system shell. Students took notes on the steps as they were demonstrated and then proceeded to practice them on their own computers, duplicating the simple knowledge base developed by the researchers. The groups were then asked to expand their knowledge bases so that they could correctly classify either a pentagon or a cylinder. This required both creating a new rule and adding to the list of possible values for a few parameters. All groups were able to complete this successfully.

In the third session, the procedures for creating the decision tree flowcharts was reviewed. Students were asked to name their favourite sports and to suggest criteria for classifying them. Categories such as "team or individual" and "indoors or outdoors" were suggested. Students then worked in their small groups to develop sports classification flowcharts using sports and categories of their choosing. A sports flowchart of the type students produced, with its component elements labelled, can be found in Appendix A. Several of the groups required some coaching from the researchers to create their charts. The most common problem was a tendency to jump from the value of one parameter to the value of the next without including the parameter names and related questions in the decision tree. Once their charts were checked by a researcher, the groups were allowed to enter the parameters and rules into the expert system shell to create a working sports knowledge base. At this stage some groups needed some guidance in defining the goal of the expert system and in the procedures for entering the parameter definitions.

Sports knowledge base creation was continued in the fourth session, which began with a question-and-answer review of the stages of knowledge base creation. The researchers engaged in more coaching and question-answering in this session. By the end of the double period, all groups had created their expert systems. Students from other groups were allowed to test out completed systems, which helped certain groups detect and correct errors in their knowledge

bases. Most bugs were caused by inconsistencies in the spelling of parameter names, which would be spelled one way in definitions and another way in rules. Others groups failed to define one or more parameters even though they were named in rules, an inconsistency which would lead to error messages when the expert system was run.

The various elements required in expert system creation (parameter definitions, prompts, etc.) were elicited from the students in a question and answer review at the start of the fifth session. The steps of debugging were also discussed, using students' experiences creating the sports knowledge base as a point of departure. Problems encountered earlier were reviewed so that students clearly saw the deficiencies in representation and/or strategy responsible for their difficulties. Strategies for remedying these problems were elicited from the students in order to reinforce their understanding of how to monitor their progress and test their knowledge bases. Students were then told that they would be developing knowledge bases to predict upcoming weather. They were questioned about the types of information that would be needed to predict weather, and the various elements were discussed. The assignment sheet was then distributed (see Appendix B). It specified the types of weather to be predicted and the criteria (parameters) to be used in the prediction rules. The task was structured to such a high degree for several reasons. Because of the relative complexity of the task for students of this age range, we felt it necessary to reduce the cognitive load of the project somewhat in order to enhance the probability that most groups could successfully complete it. By delineating the types of weather to be predicted and the criteria to be used the amount of student decisionmaking required in order to extract parameters and rules from the reference materials on weather was reduced. Less time need be spent on the question of what to include in the knowledge base, making more resources available for determining just *how* the relevant knowledge was to be transformed and integrated into the expert system. It also delimited the scope of student projects so that they did not become so complex and unwieldy that they could not be finished in the time available.

A weather prediction expert system, created by the researchers, was demonstrated to the class. A few of its parameter definitions and one of its rules were shown as models of the type of structure that students were to create. Each group was then given a packet of resource materials and told to begin the process of developing their decision tree flowcharts for their expert systems. The resource materials consisted of relevant chapters from several books on weather and weather prediction written for students at the intermediate level. They were selected to provide students with varying degrees of difficulty in the distillation of production rule

representations for prediction. The process of extracting the necessary parameters and rules from the materials was briefly modelled for the class and questions on the procedure answered. At that point the groups began their work with the materials, dividing them up so that each student had reference readings which they then began to highlight and take notes on.

Work on the weather knowledge bases continued over another seven sessions. The groups were required to develop on poster-sized flowchart sheets a full mapping of their model of the knowledge base rule hierarchies. Their completed flowchart trees had to be checked by one of the investigators before they could begin the process of entering their knowledge bases into the expert system shell. (Details of the students' work processes and experiences can be found in the Results section below.)

Both researchers closely monitored the group work on a continuing basis, taking notes in class on students' practices, strategies, and problems, and writing up their observations in detail at the end of each session. Some of the investigators' class time was of necessity devoted to coaching groups through problems that they could not overcome by themselves or (less frequently) to maintaining a work orientation in the classroom.

Several elements of the teaching and coaching techniques used were chosen based upon what recent cognitive research has suggested can best promote cognitive strategy acquisition and transfer. By limiting the initial experiences of knowledge base development to very narrow and well-known domains, we largely restricted the knowledge and strategies required for success to those that are critical to knowledge engineering. This type of initial problem restriction and simplification has been shown to facilitate problem solving and transfer (Butterfield & Nelson, 1989). Other teaching practices addressing cognitive development used in the study for which there is empirical support included: providing guidance as to what should be noticed (Butterfield & Nelson, 1989); emphasizing the salience of goals and the need to plan (Bereiter, 1981); providing detailed explanation of the strategy to be used and its conditions of applicability, as well as knowledge of its utility (Henderson, 1986; Nickerson, 1988/1989); giving instruction in production rule encoding and providing for its practice in several different contexts differing in noncritical elements so as to improve transfer (Butterfield & Nelson, 1989); encouraging active processing during representation of new tasks (Lewis & Anderson, 1985); providing instruction in and modeling of self-monitoring and flowchart evaluation for evaluating model deficiencies (Henderson, 1986); repeated review of essentials (Meyer, 1980);

and the use of coaching and fading strategies to promote self-reliance and the internalization of cognitive and metacognitive skills (Palinscar & Brown, 1984).

The Factory activity group

This group began by receiving a verbal introduction to The Factory software from a teacher. The teacher closely followed the curriculum used for working with The Factory that is detailed in *The Factory Workbook* (Blake, 1986). The modules in the workbook labelled as appropriate for grade seven and eight students were used. These usually included both on and off computer activities. Pre-computer activities involved the construction of cardboard models of factory machines and the use of them to simulate construction of a product. In the first module, students started by cutting out and assembling two machines—the punch and the striper—from xeroxed outlines. These models were then used in the second session with paper strips and cutout circles and squares to simulate the action of the factory processes of punching and striping, which were quickly mastered. The additional process of rotation was then introduced, and students worked through paper model exercises which illustrate how rotation of the product as part of the production sequence produces vertical and diagonal stripes and patterns. Students then worked in pairs using The Factory to build the simple objects shown on their worksheets using two process steps, one of which was a rotation. Next, they chose and sequenced the operations (created a "production line") that would create objects displayed by the program itself, working successfully at both the "easy" and "medium" challenge levels. In the fourth session, all groups continued to work on the computer at the medium and hard challenge levels. Some made use of the paper factory models to aid in the visualization of the effects of certain production steps. By the end of the session, some students had begun to draw their own designs which they would then attempt to create onscreen by setting up the proper production line. In the next class, students worked individually on The Factory, running timed tests to see how long it took them to make their own production lines for their products. After doing this for three different products, the teacher gathered the groups together for a discussion of their experience with The Factory. She asked them to talk about the knowledge they had gained and the strategies they had learned, and what behaviors were important for success. At the end of the session, students again worked in groups to design a production line for a very complex product assigned by the teacher.

In the third session students began using Super Factory. They started by constructing cubes which allowed them to work on paper problems involving three dimensional visualization. They then worked in the research section of Super Factory, exploring the effects of various operations. They returned to their desks to complete exercises two and four from the Super Factory workbook. Problems encountered in completing the exercises were examined in a teacher led discussion. In the following class, students worked in the design section of the program to create a factory that would output a cube which required that alternate sides in the horizontal plane be opposed by 180 degrees. This required rotation on two planes. All groups employed the cardboard cubes to help them visualize the problem.

The following session began with a discussion of the "steps to solving a problem". The class members offered suggestions and ordered them under three headings written down by the teacher: "Analyze", "Plan", and "Try". The groups then went to their computers where they worked on the "challenge" section of the program, designing production lines to make cubes of the designs modelled by the software. By the end of the class all groups had been successful at the medium level and about half had moved on to the hard level. Work on the challenges continued through the following session, by which point the groups were working at the hard and super levels. In their final class, students began work with a geopolitical simulation on ICON computers using the Ontario Educational Software Service software "Decide, your Excellency" as there were no other challenges left for them to complete with Super Factory.

The project group

Students in this class divided into six groups of four. Group membership was self-selected. The class was led by both the science and industrial arts teachers. Each group was given photocopied instructions for making a number of weather monitoring instruments: an anemometer for wind speed; a wind vane for wind direction; a hygrometer for relative humidity; a barometer for air pressure; and a rain guage for precipitation. The groups' first task was to make a list of materials needed and to decide which student would make which instrument within each group. In the following session, students were instructed to set up and start a daily journal which would be used to write down the problems that had to be solved that day, and the plans that would allow them to solve the problems. Each student was also told to prepare four small graphs for wind speed and direction, precipitation, air pressure, and relative humidity, that would be used to record daily instrument readings. The students were assigned the task of

deciding what type of graph was most appropriate for each data type and how it should be scaled. They were also told to begin research logs in which they would record the work done in every class. Some students went to a nearby hardware store to purchase required materials.

Students spent the third session building their instrument stations and developing their graphs. They were also told to make large graphs for each group, one for each type of data. Some of this work was completed in the fourth class. Most students managed to take their instruments outside to commence their observations in this session. Instruments were largely finished in the fifth session. Students were instructed to develop calendars for the daily recording of their observations. The next few classes were spent collecting, recording and graphing instrument data, and studying written materials from the library on how to predict weather. Students were told to write short predictions in their journals for the next day's weather, and to be sure to complete their research logs. The science teacher told the groups that she would be checking their graphs. Work on journals, calendars, graphing, and prediction continued over the remaining three sessions.

Results

Quantitative Analysis

Analyses of covariance were performed on the posttest scores from each of the three Differential Aptitude Test (DAT) scales (verbal reasoning, spatial reasoning, and abstract reasoning) using the matching DAT pretest scores as covariates. No significant differences were found between groups at the .05 level. Analyses of variance were undertaken for each of the remaining post-tests—the Arlin Test of Formal Reasoning, the Student Thinking About Problem Solving Scale, and the four transfer measures. There were no significant differences in mean group scores for any of these measures either. The univariate *F* scores for each test, along with the posttest means and ANOVA results for the three groups on each measure, are shown in Table 2 (see next page). Table 1 gives the pretest means by group.

TABLE 1

Mean pretest scores by group

Test	Group		
	Expert systems	Factory activity	Project
DAT—Abstract reasoning			
M	30.23	31.22	30.34
SD	6.66	6.28	6.64
N	22	22	23
DAT—Spatial reasoning			
M	33.78	29.91	27.09
SD	10.38	8.64	7.27
N	23	23	23
DAT—Verbal reasoning			
M	17.68	17.22	15.74
SD	8.90	6.26	5.10
N	22	22	23

TABLE 2
Mean posttest scores by group

Test	Group			*df*	*F*
	Expert systems	Factory activity	Project		
DAT—Abstract reasoning					
M	32.36	32.18	31.22	2, 63	0.12
SD	5.99	6.75	7.95		
N	22	22	22		
DAT—Spatial reasoning					
M	36.57	34.74	33.19	2, 63	0.65
SD	10.18	10.09	8.73		
N	21	23	22		
DAT—Verbal reasoning					
M	20.59	19.82	17.95	2, 63	0.56
SD	10.58	7.14	6.88		
N	22	22	22		
Arlin Test of Formal Reasoning					
M	14.74	13.18	12.25	2, 62	2.02
SD	4.84	3.96	3.29		
N	23	22	20		
STAPSS					
M	129.5	131.9	126.3	2, 61	0.51
SD	15.82	20.72	16.42		
N	22	22	20		
Transfer task 1					
M	24.95	22.69	22.30	2, 64	0.936
SD	5.52	6.71	8.04		
N	21	23	23		
Transfer task 2					
M	5.57	4.91	5.39	2, 64	1.14
SD	0.98	1.81	1.56		
N	21	23	23		
Transfer task 3					
M	5.29	5.61	5.00	2, 64	0.36
SD	2.15	2.59	2.47		
N	21	23	23		
Transfer task 4					
M	5.28	4.69	3.96	2, 64	1.38
SD	3.07	2.78	2.08		
N	21	23	23		

All groups showed a strong pre-post gain in their DAT scores on each of the three scales. These gains, made over a three month period, were equal to or greater than those the scale norms indicate should be expected after six months for students at that grade level.

As our observations of the students developing expert systems suggested that some students were more able to cope with the demands of the task than others (see below), we conducted a further analysis to address the question of the extent to which prior abstract reasoning ability interacted with the relative benefits of the various treatments. It was hypothesized that students of higher abstract reasoning ability were more likely to make transferable gains in the skills required in knowledge engineering because with their more mature initial abilities they were more likely to have a zone of proximal development conducive to mastering these skills. Analyses of variance for the posttests were conducted for all students who scored above the grand median on the DAT Abstract Reasoning pretest. It was found that the group means were significantly different for three of the posttests: the Arlin, the DAT verbal reasoning scale, and the fourth transfer task, with the expert system group scoring the highest in all cases (see Table 3, next page).

TABLE 3
Mean posttest scores by group for students above grand median on DAT abstract reasoning pretest

Test	Group			*df*	*F*
	Expert systems	Factory activity	Project		
DAT—Abstract reasoning					
M	37	35.16	37.1	2, 27	1.37
SD	2.92	2.62	3.60		
N	8	12	10		
DAT—Spatial reasoning					
M	44.38	39.23	40.2	2, 28	1.06
SD	8.14	9.37	5.71		
N	8	13	10		
DAT—Verbal reasoning					
M	30.37	21.91	21.20	2, 27	4.04*
SD	9.77	5.80	7.31		
N	8	12	10		
Arlin Test of Formal Reasoning					
M	19.13	15.00	12.00	2, 25	7.99**
SD	4.26	2.92	3.78		
N	8	12	8		
STAPSS					
M	140.62	135.17	122.78	2, 26	2.07
SD	11.46	20.44	21.3		
N	8	12	9		
Transfer task 1					
M	26.33	23.15	24.00	2, 27	0.39
SD	4.72	7.39	8.17		
N	6	13	11		
Transfer task 2					
M	6.00	5.54	5.36	2, 27	0.57
SD	0.00	1.20	1.43		
N	6	13	11		
Transfer task 3					
M	6.33	5.92	4.45	2, 27	1.43
SD	1.63	2.63	2.80		
N	6	13	11		
Transfer task 4					
M	7.83	6.30	4.73	2, 27	7.37**
SD	1.72	1.49	1.74		
N	6	13	11		

*$p<.05$ **$p<.005$

Qualitative observations

Expert Systems group

During the last stages of the training phase, when students in the expert systems group were developing their sports knowledge bases, several deficiencies in knowledge representation and strategy execution became evident in the work of some students. In constructing their knowledge base flowcharts, many failed to include either the goal parameter or the names of other parameters—only the value of the parameters would be entered. Prompts would also be missing. Usually the groups needed some prompting to draw their attention to the functional goal of the expert system, and coaching was often required before they could select the right strategies for developing and mapping rules. Leading questions meant to encourage metacognitive thinking, such as "What do you think you need to know to develop your rules?" and "What steps do you have to take next?" would sometimes be sufficient to get students to develop and apply effective procedures. Other individuals would need more directive coaching and modelling of the desirable strategies before they could proceed successfully on their own.

Entry of the rules and parameter definitions into the PC Easy shell brought with it its own set of difficulties. Most groups needed help in mastering the multi-step process of defining the goal and other parameters and entering the prompts associated with each parameter. In the review that followed the sports knowledge base creation, students proved very able to describe the components necessary to knowledge base mapping, and could outline the steps needed to develop an expert system using the shell.

Some groups were unclear as they began their work on the weather assignment how to use the printed materials they had been given for reference. They knew what the general goal of the project was, but did not immediately grasp the salience of the reference texts to the task of rule development. It was unclear where they thought that they were going to get the information needed to allow prediction rule development —they possibly thought that they could come up with the rules themselves as they had with the sports knowledge base. Once the link between tasks and materials was made clear to them, they started to take notes on the texts in order to extract rules and parameters. Those students working with book extracts that presented prediction rules either in tabular form or fairly explicitly in their texts had little difficulty developing lists of rules, although some had to be guided to using the IF-THEN format for rule

representation. Others, who were working with more conceptually complex chapters on weather that required a greater transformation in students' knowledge representation for prediction rules to be successfully extracted, had much more difficulty and frequently had to see the process being modelled with their materials before they could develop rules autonomously. Questioning and prompting support strategies alone were not usually effective in these situations.

The construction of the decision tree flowcharts proved problematic for some groups. Various kinds of difficulties were encountered at this stage. Many students found it difficult to recall the components—parameters, prompts, and values—that needed to be entered into the chart and how to represent them (boxes around parameter names and prompts, links to values, and so on). Once again the use of leading questions was enough to help some groups realize what components were needed and the procedures to follow, while others needed more direct modelling and instruction before they could be independently successful. The two most pervasive deficiencies in the students' mappings of their knowledge bases were indicative of the incompleteness and lack of integration of their mental representations of their collections of rules. The first of these was a failure to include parameter names in the chart mappings. Students would enter a prompt question, such as "What type of clouds are visible?", and link it directly to the possible values (cirrus, cumulus, etc.) and then to the next prompt needed by the rule without labelling the prompt with its associated parameter name. (Readers are referred again to Appendix A to see the correct flow chart structure). Stated verbally, the incorrect representation would be analogous to saying, for example, "If there are cirrus clouds and northwest winds and it is cool... then rain is likely" rather than the correct "If the cloudtype is cirrus and the wind direction is northwest and the temperature is cool... then the prediction is rain". This error in representation is very understandable, for the incomplete version (without parameter names) is of course much more typical of the representations that are used in the thinking and discourse of everyday life. In normal conversation, the parameter referents for the values are automatically inferred, and so need no explicit articulation. The expert system shell however is not this intelligent and must be shown the explicit link if it is to work.

The second most common error revealed a different deficiency in the domain integration of the students who exhibited it. There was a common tendency to map all rules independently, even when they could share the same first parameter prompt and branch off from this common base. For example, several rules might have as their first clause "If cloud type is X...". Those

students who understood the relationships between the rules in the knowledge base would diagram these rules in such a way that they shared a common root parameter ("cloud type") and prompt ("What types of clouds are there?") and would be branched off according to their differing values for that parameter (cirrus, etc.). But many students would map the rules independently, placing a different parameter box at the top of the chart for each of the four or five cloud types. Each occurrence of a parameter (e.g. cloud type) was sometimes given a different name when it was entered into the charts, unnecessarily increasing the number of parameters that had to be defined. Instances of this localization of focus could also be seen at another stage of the expert system development process as well, when different rules referencing what in fact were different values of the same parameter used different names for theparameter. Taken together, these observations suggest that these students developed each rule independently, without any consideration of their relation' ip to other rules.

Two groups had almost the opposite difficulty. Having been trained initially in developing knowledge bases that began hierarchically with only one root, they were uncertain how to map out rules that did not share the same parameter and prompt at their base. Once informed that they need not start all rules from the same root they had no difficulty in proceeding.

A less common deficiency observed was an inability to differentiate between the current value of a weather factor—say, barometric pressure—and its *delta* or change value—change in barometric pressure (rising, etc.). These were two different types of parameters, but students would occasionally confuse the two or attempt to collapse them together by entering the values associated with one (such as changes in clouds) into the parameter definition for the other (cloud type).

Socratic questioning and coaching was nearly always effective in getting groups and individuals "unstuck" from these problems, although in a few instances repeated assistance was required. A class review in the third weather session of rule development using parameters and values and the mapping of rules in a tree diagram also proved helpful.

A second set of difficulties emerged as the groups began to enter their knowledge bases into the expert system shell. Some of these problems were syntactical in nature; the shell required that certain conventions be strictly adhered to in the entry of rule and parameter definitions. It was critical that the spellings of parameter names in rules be identical to those found in their

definitions, or the shell would assume that a new parameter was in need of definition and would pop up a window for that purpose. This would inevitably confuse students at first, but once they understood the underlying cause they would correct the misspelled word(s) and proceed on. Unfortunately the misspelling of parameter *values* in a rule was not so easily caught as the shell would not provide any warning about it during rule entry. The existence of the discrepancy would only emerge during a consultation, when the selection of that value by the user in response to a prompt would not fire the appropriate rule, and the expert system would not reach any conclusion. The debugging of these fairly common errors, required examining each rule in turn to find the one that contained the misspelling and correcting it in the rule definition. This in turn often required reference to parameter definitions and value lists so that spelling comparisons could be made since students would not usually have memorized all the values for all the parameters. Alternatively the misspelling of a parameter value might have occurred in the parameter definition stage, and the spellings in the rules be correct. This was a little easier to detect and localize since the misspelling would then appear as one of the possible choices in response to the parameter's prompt during a consultation.

Syntactical errors were also common, and were a source of some frustration to most students, as the debugging required to fix them was thought to be onerous and rather uninteresting, and some students would start engaging in different off-task activities to avoid dealing with it. Most groups had to have this debugging process modelled and explained to them before they were able to engage in it independently.

There was another type of strategic deficiency that revealed itself as students entered and debugged their knowledge bases. When defining parameters, students would sometimes only enter the value that the parameter took in one rule, ignoring its value in other rules. To overcome a localized focus on one rule at a time in defining parameters and to help students integrate the rules into a coherent mental representation of the whole knowledge base, groups were encouraged to look across the descending rule chains so as to discover all the values required for the parameter when defining it. If certain values were left out, the rules containing those values could not be fired in a consultation, and the values could not appear as possible choices for user response to that parameter's prompt.

Observations of the students' interactions over the course of the study suggested that groups of three were not the ideal size for expert system projects. This grouping presented no difficulties

while students were using the reference materials to develop rules, since there were enough materials for each student to work independently. But during the creation of the large flowchart, only one student at a time could be translating their written rules into flowchart form, with perhaps a second actually doing the drawing. The third student, with little to do, would often wander around distracting other groups. Similarly during the entry of the knowledge base into the shell, two students might work collaboratively from their chart, one at the keyboard and the other with the chart, but rarely would all three group members be involved. In a few of the groups there was a definite tendency for one person to dominate the group's activity. These individuals would have drawn up the entire chart and taken the entire responsibility for the entry and debugging of the knowledge base had it not been for the intervention of the researchers.

In general, the students' levels of motivation seemed to vary from session to session and student to student. On some days there was very little disruptive or off-task activity; on others, (usually in the debugging stage), upwards of a third of the students could be wandering around or otherwise off-task. Because both researchers were frequently engaged in coaching or checking group efforts, class control was not as tight at these times as it might normally be. By the end of the sessions, all but two of the groups had working systems. (One of the incomplete groups had lost a great deal of work due to a hard disk crash; the other saved their expert system file incorrectly. Both had had to restart entry from the beginning.) The groups had been able to create functioning knowledge bases of moderate complexity, varying from 6 to 20 rules and 10 to 16 parameters in size. (Appendix C shows a representative section of the tree chart for one of the student knowledge bases.)

Students were pleased to see their own systems working, and seemed to enjoy demonstrating them to their peers from other groups and especially to those who had been in the Factory activity and project sections, who tried out their systems at the end of the study.

Factory activity group

All of the students using The Factory and Super Factory were able to successfully meet the challenges presented by both programs, and most could do so even at the highest level of difficulty the programs offered. Initially most students had difficulty understanding and visualizing the rotations required in the more advanced production sequences. (Object could be rotated by multiples of 45°, but only counterclockwise rotation was permitted.) They had little

prior understanding of the measurement of the degree of rotation. Virtually all of the students' design failures when creating factory lines to produce a given object were due to incorrect rotational operations—usually when the required rotation was by some uneven multiple of 45° such as 135°. This problem occurred at all levels of program difficulty. For some, work with the paper models helped them to understand the concepts involved more fully. Others commented that they thought the paper models were unnecessary and preferred to work only on the computer. Because students were limited to eight total operations to generate the desired product, those who grasped the notion of arranging their production steps so that the fewest rotations possible would be necessary had greater success with the most complex object challenges. The initial tendency was usually to attempt the most obvious operations first (for example, the creation of a bold stripe with 3 punched holes). As a result, students would often "run out" of operational steps, or alternatively they might succeed in performing the required number of operations but be unable to either orient these correctly or to reorient the product for final display. These difficulties were most acute in the case of the challenges that the students created for themselves in both programs, which went beyond the difficulty level of even the toughest computer-generated challenges. These problems were usually mastered as students came to see that they had to shift from a "what next?" strategy to more comprehensive planning of the entire series of operations if they were to be successful.

Students would generally design a portion of their factory operation sequence collaboratively with their partners, and then process the product on the computer to debug their design. Interestingly, some of the students who appeared to have the greatest difficulty with the programs in their first encounters proved to be the most successful in the later, more difficult challenges. Nearly all students quickly mastered the programs' simple interfaces; there appeared to be no operational difficulties beyond some slight impatience with the slowness of the PCjr's graphics refresh rates. The students' apparent levels of intrinsic motivation were very high throughout the period of The Factory use. Very little off-task activity was observed, and several students were observed working through the recess period that followed the class on a few occasions.

When students first began work with the Super Factory, most exhibited considerable resistance to engaging in the task of three dimensional visualization and rotation that its tasks demanded. The problem of visualizing rotation in three planes was felt to be overwhelmingly difficult by some. Even once they understood what steps were required to obtain the proper positioning,

students would often lose track of the cube's position at some point in the designing process and erroneously stamp "over" already created sides. But with some practice, they were able to keep track of positional movement in all three dimensions, at which point their progress speeded up, and their motivation returned to its previous high level. By the last session, a number of students had developed very logical and successful strategies for manipulating the cube, often using fewer factory steps than the number used in the program's own solutions.

Project group

Students in the project group had no significant difficulty building their weather instrument stations. Most of the teams completed both their small and large graphs and recorded the daily data gathered from the stations on them, although the process of chart development was confusing to some who had had limited prior experience in graphing. Students generally maintained a high level of interest in their work over the sessions, although a few individuals were repeatedly off task. Student interest was heightened by seeing their instruments working. When told to use their data to make predictions using forecasting rules found in their science text, most did not know how to proceed, despite the fact that instructions for doing this were given in their written materials. When the teacher inquired as to how they had arrived at their written predictions, they indicated that they had taken then from the newspaper or merely made a guess. Some students were in fact copying their neighbour's work. After explicit instruction and modelling of the process by the teacher, students did start to develop their own predictions based from their instrument readings and the rules in their texts.

Discussion

The observational data clearly indicate that creating a working expert system was the most cognitively demanding of the three tasks. It took repeated exposure to modelling, individual practice, and monitoring and coaching by the researchers before the average student was able to successfully extract salient knowledge from the texts and actively transform its representation from a declarative form to a procedural one. Initial success at this task was typically only partial and piecemeal; a student might be able to develop individual production rules for prediction, but could not fully integrate the various rules into an integrated, coherent procedural representation of the domain. This was reflected in the difficulty most groups had in interrelating the rules in their initial diagramming attempts. It will be recalled that it was common for parameters shared by various rules to be differently named in each rule, which is indicative of the students' ignorance of the shared predictive factors. Once they were made aware of the existence of these relationships, however, most students were able to modify their charts appropriately to develop an adequate representation of their domains.

It was clear that the process of knowledge base development forced students in the expert systems group to engage the subject matter on weather in a deeper and more meaningful way than did students in the project group. Only by understanding and mapping the domain as a coherent network of propositions and rules could they develop knowledge bases that were reasonably accurate and free of crippling flaws. Students in the project group could take a much less demanding route to making their predictions; they merely had to consult a list of pre-existing rules to see which ones applied to their data, and thus need not perform the cognitive processing necessary for deep domain understanding. (However other skills such as those relating to gathering and plotting data were likely furthered by their project work.)

Why, then, were there no significant differences found over the whole sample favouring the expert system group in the various measures of transformative skill and general cognitive abilities? There are several possible explanations. The exercise of knowledge engineering, at least as implemented in this study, may simply be ineffective in developing any transferable skills. Alternatively, the measures may have been too wide-ranging. The Differential Aptitude measures employed assessed three broad domains of cognitive ability—abstract reasoning, verbal reasoning, and spatial representation and transformation. It may be that exposure to 25

or so hours of work in expert systems development may be inadequate as a means of fostering significant growth in a such a broad range of skills, most of which are much more general, a far distance in terms of any transfer metric from the specific strategies and transformations students employed in their work. The same argument applies in the case of the Arlin Test of Formal Reasoning, another very general measure of cognitive maturity. While two of the Piagetian formal schemes assessed, correlational reasoning and combinatorial reasoning, were central to the expert systems task, several others measured were not. And even these two forms of thinking were not unique to the expert system group, as they were employed by those solving problems in The Factory and Super Factory as well.

A similar line of reasoning can be applied to the metacognitive self-report measure results. Given the limited scale of the treatment, it may be overly optimistic to reasonably expect any global changes in self-reported metacognitive functioning. Finally, with reference to the transfer measures, it may be that the far transfer tasks were too far removed from what was practised in the expert systems work to show significant gains, as they required different forms and strategies of knowledge transformation and classification.

However, the study's other results offer grounds for contending that there exists a more plausible alternative to the hypotheses of treatment ineffectiveness or the overly broad focus of the assessment tools. It will be recalled that among those students over the median on the abstract reasoning pretest, the mean scores for those in the expert systems group were significantly greater on three of the seven posttests (the Arlin, the DAT verbal reasoning scale, and the fourth transfer task). This suggests that the expert systems task as it was undertaken in the experimental context was of such difficulty that only those students bringing to it above average abstract reasoning skills for their grade level made appreciable cognitive gains. Put another way, the task as it was taught and assigned may have been beyond the zone of proximal development for the less able students. The observational data offers some support for this interpretation. Certain students had great difficulty mastering the transformative strategies the task required, and could achieve partial success only with repeated coaching and scaffolding by the researchers and other students, whereas others needed very little external support to succeed. One can only speculate given the present data as to whether a more extensive cycle of teaching, practicing and coaching would have brought the lower-scoring students up to a level of skill in the needed strategies that would result in a measurable gain in broadly applicable cognitive abilities. It may be that no amount of expert systems work at this age level would provide such

benefit, although the fact that nearly all of the expert systems students were able by the end of their sessions to carry out most of the steps involved in building simple systems suggests that this may not be the case.

It may also be that no relative advantages were found for the expert system treatment sample as a whole because *all* groups made above-normal gains in reasoning ability due to the fact that all three received treatments that were in some way enriched relative to traditional textbook-based teaching. It was noted earlier that students in all three groups showed pretest-postest gains in the three Differential Aptitude Test scales at a level that the test norms indicate would usually occur only after six months—over twice the testing interval. In any further comparative research, it may be advisable to subject one group to a more traditional curriculum and pedagogy.

The expert system project

Our observations suggest that there were elements of both the expert systems project and the PC Easy expert systems shell that may have increased the difficulty of the knowledge engineering task. Using a hierarchical flowchart structure to map out the knowledge base parameters and rules may not have been the most effective aid for helping students develop an integrated and proceduralized representation of the weather prediction domain. It will be recalled that some groups used different names for the same parameter in the different rules of their flowcharts, suggesting that the charting process did not help these students move from an isolated to an integrated representation of the knowledge base rules. The apparent inadequacy of hierarchical mapping as a cognitive aid in the weather task is probably due to the fact that such a form is not isomorphic to the domain's structure. Unlike, say, rules of taxonomic classification, the rules used in weather prediction are largely nonhierarchical and nonbranching in form. To use a statistical metaphor, they constitute a fully cross-factored rather than a nested design, since each prediction rule is based upon values from set of factors or parameters shared by nearly all the other rules, and differs from the others only in the values assigned one or more of the parameters. For example, virtually all prediction rules take into account the current cloud conditions, the wind direction, and the direction of barometric pressure movement. A more isomorphic mapping of a fully cross-factored domain could be achieved by using a tabular form, which would have the virtue of more closely relating the various parameters to the rules in which they were embedded. As such, it would likely be of greater value in helping students

internalize a more coherent and integrated domain representation. A suggested format using hypothetical weather prediction rules is shown in Table 4. Parameter names label the columns, and rule names label the rows. Cells are given the values for the parameters appropriate for the rule on that row, with irrelevant parameters for a given rule having blank cells. Rules are read from left to right.

TABLE 4

Tabular knowledge base mapping

Rules	Parameters				
	Cloud type	Pressure change	Wind direction	....	Prediction
Rule 1	stratus	falling	southwest	....	steady rain
Rule 2	clear sky	rising	north	....	sunny, cool
Rule 3	....	....	....	....	....

By making more visually immediate the relationships between parameters, values, and rules in a factored domain model, this form of mapping should facilitate a more complete integration of domain knowledge and so avert any tendency to use duplicate parameter names in rule entry.

Features of the expert system shell

Many of the frustrations that students experienced in entering and debugging their parameter and rule definitions could be ameliorated by improvements in the expert system shell environment. As mentioned earlier, the PC Easy shell was not capable of checking the values for parameters as they were being entered into rules against the value lists entered into the parameter definitions. Any errors in spelling could only be detected when a consultation did not respond correctly to a value selection in a consultation. When upwards of twenty rules, each having several clauses, were entered and then tested, tracing the rule that caused the problem or problems (and often there was more than one) proved tedious and time-consuming, and usually lowered students' enthusiasm for their task. Expert system project work is lengthier and more demanding than typical classroom activities even without these complications. Unlike The Factory activities, in which success at some level (and its consequent satisfactions) are rapidly

achieved, the gratification that can come from creating a working expert system may seem so distant as to lead students to pursue other more immediate rewards. In order to prevent a flagging of student interest, expert system projects for younger students should make use of shells that provide more complete real-time checks of syntax and structural consistency during rule and parameter entry. Better rule tracing facilities to assist the debugging process would also be desirable. While PC Easy did provide a tracing option, it was too complex for young students to use effectively.

Our observations suggest that knowledge base debugging can be greatly facilitated within any shell environment by entering and testing rules in an iterative cycle, rather than waiting until all rules have been entered before checking for problems. Groups in the present study that used this iterative procedure found it much easier to trace and correct misspelled parameter values and names and incomplete branches, since only one rule rather than the whole list had to be checked against parameter definitions and the chart if an error was found. Since there would usually be several errors made in the course of entering the entire set of rules, disentangling what problem caused a runtime error often proved frustrating to those who left the task of debugging to the end. A further advantage of the iterative method lies in its greater motivational potential. By allowing students to achieve at least some partial success more quickly, it may help to maintain student enthusiasm at a high level throughout the entry and debugging phase of their projects.

Other difficulties emerged in the knowledge base testing and debugging processes. Students would sometimes have difficulty locating the appropriate rules and parameters on their large charts when debugging their work. The charts' large size made them awkward to work with when sitting at a keyboard. The use of tabular domain mapping would solve the size problem to some extent, and (as mentioned earlier) its format would facilitate the crosschecking of parameters, values, and rules. Another source of student frustration lay in the cumbersome procedures required to examine the rule and parameter definitions that had already been entered into the shell for syntactical and structural errors. Several keystrokes would be required to get to the desired screens. Access to definitions would be greatly facilitated by providing the students with access to a printer, perhaps via a network, so that they can print out all entered rule and parameter definitions for checking. (Unfortunately this was not possible in the present study.)

Group interaction and motivation

Our results indicate that having students work in groups of three for expert systems development can result in a disparity of effort and accomplishment within groups. While any number of students can research rules in reference materials, the processes of chart creation and entry of the knowledge base into the shell are constrained to one location, which seemed to make it awkward for more than two people to participate. Integration of the rules, to the extent that it occurred, took place in two stages; one pair would merge their findings, and the result would then be amalgamated with the third student's information. This typically left one or other of the students "out of the loop" most of the time. Chart creation was also usually coordinated between two individuals, with one often dictating the rules and the other drawing and labelling the diagram. A similar pattern held for shell entry; one student would extract and dictate the required information from the chart and another would select the correct entry screen and key in the data. There was a widespread tendency for the third party to be off task or out of the area during both of these phases. It seems likely that the use of pairs rather than triads would induce higher levels of involvement since no student would find themself in situations in which there were very limited opportunities to collaborate in the task. This was certainly the case with the Factory activity groups, in which collaborative involvement in pair activity remained high throughout the experiment.

Despite the limitations of the triadic arrangement, there was often a high level of cooperation exhibited within groups. This was most evident when some problem arose in chart creation or knowledge base entry and debugging. Students would actively discuss the potential sources of error, making suggestions and commenting on or trying out the suggestions of others. Two girls in particular became very proficient at debugging and helped students in other groups. Very similar interactions were common in the Factory activity groups as well, as they worked to solve factory problems. Because students in the project groups worked more independently on their charts and diaries, less cooperative problem-solving was seen in their activities, although occasionally a student would ask another for assistance if he or she was stuck.

While student interest levels tended to fluctuate in the expert systems groups, motivation remained at a more sustained plateau in the Factory activity class. The reasons were readily apparent, and some have already been mentioned. The key element appeared to be the length of

the reinforcement cycle. Students working with The Factory had to deal with discrete, small problems that could be solved in a matter of minutes, after which they quickly received feedback about the correctness of their efforts, and (provided their solution was correct) could then enjoy the intrinsic satisfaction of accomplishment. If errors were encountered, they did not cause a great deal of frustration, since students knew that they could be solved in relatively short order. The situation facing the expert systems groups was quite the opposite. Knowledge of success and the attendant satisfaction was perceived to be a very long way off, since the only meaningful feedback indicative of achievement would come when the expert system actually worked, which might be days or weeks away. The task thus pushed many students to the limit of their adolescent capacity for tolerating delayed gratification. Also contributing to the vacillation of student motivation in the expert systems group was the frustration caused by the various factors that inhibited the easy entry, testing, and debugging of knowledge bases. Still, the majority of students maintained enough interest in their projects to experience some satisfaction in seeing their systems working at the end of the unit.

The participants in the instrument project groups fell somewhere between those in the other two treatments in terms of their enthusiasm for their work. Initially they were very excited about developing their stations and making predictions, but by the last few sessions their interest had flagged somewhat.

Summary and conclusions

Students in all three conditions were largely successful in completing their assigned tasks or activities. The expert systems groups were able to create functioning knowledge bases of moderate complexity, varying from 6 to 20 rules and 10 to 16 parameters in size. Problems were encountered in three stages of the process. Because the first knowledge bases developed in the instructional period were mapped completely hierarchically, students encountered difficulty in plotting rules that made use of many of the same parameters. Other strategic deficiencies became evident when students attempted to convert the mapped production rules on their charts into the syntactical structures needed to enter them into the expert system shell. Some students were initially not able to see that they were actually employing the same parameter in different rules but applying different values to it. As a result they would define the same parameter more than once, assigning a different name and value to it in each rule. Others would fail to rigourously translate their rules into production syntax, and so would include only the values and not the names of the parameters in the rules. Both of these problems seemed indicative of an inability on the part of some of the students to develop a differentiated and integrated perspective on the entire production system and its internal relations. The use of a tabular form of knowledge mapping for nonhierarchical, cross-factored domains such as weather prediction may help to ease many of these difficulties in representation and translation, as it is more congruent to the conceptual structure of the domain. The final major source of student error was syntactical; it arose from the shell's requirements for precision in spelling and punctuation in data entry and operational problems in tracing bugs. The PC Easy expert system shell offered minimal automatic structural checking and lacked easily used tracing and debugging tools to assist in locating these problems. Our experience argues for their inclusion in any shell to be used with students of this age or younger.

No firm conclusions can safely be made about the lack of significant differences between the three groups on the dependent measures. A possible explanation that cannot be fully discounted is that the expert system work was ineffectual in promoting the broader forms of cognitive growth that these instruments addressed. But it seems to us that two secondary findings—significant test differences favouring the expert systems group on three of the measures when limiting the analysis to students in the overall sample who were above the grand median on the abstract reasoning pretest, and the strong gains in verbal, abstract, and spatial reasoning made

by all three groups over the treatment period—suggest two more optimistic interpretations. Given the difficulty of the expert systems project for certain children, it may be that the knowledge base development task as assigned was either too demanding or the time and teaching devoted to it too short for it to be of significant benefit to students of below average abstract reasoning skills. Alternatively, the intensive and problem oriented nature of all three treatments may have led to above-normal skill growth in all three groups, diminishing potential intergroup differences. In order to further comparative anlysis, future studies would be advised to more closely follow a traditional curriculum in the control group.

In sum, our results provide partial support for the viability and efficacy of knowledge base creation as an instructional strategy for fostering cognitive development at the grade eight level. It may be that knowledge engineering can only be fruitfully employed as an enrichment activity for those pre-high school children who possess above average abstract reasoning ability. But it would be premature to discount the value of expert system activity for the average intermediate level student yet. We have suggested several alternative procedures and changes in expert shell design that could potentially enhance its viability for this population. Assigning shorter and less complex knowledge base development tasks may be another way to increase the amount of skill acquisition and transfer. Before we can answer these questions, we will need to undertake a series of investigations in which expert system projects are integrated into curriculum and pedagogical practice in a number of ways so that their possibilities and limitations can be more thoroughly understood.

References

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

Arlin, P. K. (1984). *Arlin Test of Formal Reasoning.* East Aurora, NY: Slosson.

Armour-Thomas, E., & Haynes, N. M. (1988). Assessment of metacognition in problem solving. *Journal of instructional psychology, 15*(1), 87-93.

Bennett, G. K., Seashore, H. G., & Wesman, A. G. (1988). *Differential Aptitude Tests* (Canadian ed.). Toronto: Harcourt Brace Jovanovich and The Psychological Corporation.

Bereiter, C. (1981). Development in writing. In C. H. Frederiksen & J. F. Dominic (Eds.), *Writing: The nature, development, and teaching of written communication.* Hillsdale, NJ: Erlbaum.

Bereiter, C., & Scardamalia, M. (1986). Educational relevance of the study of expertise. *Interchange, 17*(2), 10-19.

Blake, P. (1986). *The Factory Workbook.* Pleasantville, NY: Sunburst Communications.

Brainin, S.S. (1985-86). Mediating learning: Pedagogic issues in the improvement of cognitive function. *Review of Research in Education, 12*, 122-155.

Bransford, J., Sherwood, R., Vye, N., & Rieser, J. (1986). Teaching thinking and problem solving. *American Psychologist,41*, 1078-1089.

Brown, A. L., Bransford, J. D., Ferrara, R. A., & Campione, J. C. (1983). Learning, remembering, and understanding. In J. H. Flavell & E. M. Markman (Eds.), *Handbook of child psychology: Vol. 1. Cognitive Development* (pp. 77-166). New York: Wiley.

Butterfield, E. C., & Nelson, G. D. (1989). Theory and practice of teaching for transfer. *Educational Technology Research & Development, 37*(3), 5-38.

Clements, D. H. (1985). Research on Logo in education: Is the turtle slow but steady, or not even in the race? *Computers in the Schools, 2*(2/3), 55-71.

Collins, A., Brown, J.S., & Newman, S. (1988). The new apprenticeship: Teaching students the craft of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Cognition and instruction: Issues and agendas.* Hillsdale, NJ: Erlbaum.

Day, J. D., & Hall, L. K. (1988). Intelligence-related differences in learning and transfer and enhancement of transfer among mentally retarded persons. *American Journal on Mental Retardation, 93*, 125-137.

Derry, S. J., & Murphy, D. A. (1986). Designing systems that train learning ability: From theory to practice. *Review of Educational Research, 56*(1), 1-39.

diSessa, A. A. (1988). What will it mean to be "Educated" in 2020? In R. S. Nickerson & P. P. Zodhiates (Eds.), *Technology in Education: Looking Towards 2020*. Hillsdale, NJ: Erlbaum.

Dreyfus, H. L. & Dreyfus, S. E. (1986). *Mind over machine*. New York: The Free Press.

Glaser, R. (1984). Education and thinking: The role of knowledge. *American Psychologist, 39*(2), 93-104.

Greeno, J. G., & Simon, H. A. (1988). Problem solving and reasoning. In R. C. Atkinson, R. Herrstein, G. Lindzey, & R. D. Luce (Eds.), *Steven's handbook of experimental psychology* (2nd ed.). New York: Wiley.

Henderson, R.W. (1986). Self-regulated learning: Implications for the design of instructional media. *Contemporary Educational Psychology, 11*, 405-427.

Lesgold, A. (1988). Problem solving. In R. J. Sternberg & E.E. Smith (Eds.), *The psychology of human thought* (pp. 188-213). Cambridge, England: Cambridge University Press.

Lewis, M., & Anderson, J. R. (1985). Discrimination of operator schemata in problem solving: Learning from examples. *Cognitive Psychology, 17*, 26-65.

Lippert, R. (1988, April). *Linking recent research in cognitive science and problem solving to instructional practice: New possibilities.* Paper presented at the annual meeting of the American Educational Research Association, New Orleans.

Littlefield, J., Delclos, V. R., Bransford, J. D., Clayton, K. N., & Franks, J. J. (1989). Some prerequisites for teaching thinking: Methodological issues in the study of Logo programming. *Cognition and Instruction, 6*(4), 331-366.

Meichenbaum, D. (1986). Metacognitive methods of instruction: Current status and future prospects. *Special Services in the Schools, 3*(1), 23-32.

Meyer, R. E. (1980). Elaboration techniques that increase the meaningfulness of technical text: An experimental test of the learning strategy hypothesis. *Journal of Educational Psychology, 6*, 770-784.

Mitchell, J. V. (Ed.). (1985). *The ninth mental measurement yearbook*. Lincoln, NE: University of Nebraska Press.

Nickerson, R. S. (1988/1989). On improving thinking through instruction. *Review of Research in Education, 15*, 3-57.

Palinscar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension fostering and comprehension monitoring activities. *Cognition and Instruction, 1*(2), 117-175.

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

Pea, R. (1988). Putting knowledge to use. In R. S. Nickerson & P. P. Zodhiates (Eds.), *Technology in Education: Looking Towards 2020* (pp. 169-212). Hillsdale, NJ: Erlbaum.

Personal Consultant Easy. (1986). [Computer program]. Austin: Texas Instruments Incorporated.

Scardamalia, M., & Bereiter, C. (October, 1989). *Schools as knowledge-building communities*. Paper presented at the Workshop on Development and Learning Environments, University of Tel Aviv, Tel Aviv, Israel.

Scardamalia, M., Bereiter, C., McLean, R. S., Swallow, J., & Woodruff, E. (1989). Computer-supported intentional learning environments. *Journal of Educational Computing Research, 5*(1), 51-68.

Shuell, T. J. (1986). Cognitive conceptions of learning. *Review of Educational Research, 56*(4), 411-436.

Soloway, E. (1988). It's 2020: Do you know what your children are learning in programming class? In R. S. Nickerson & P. P. Zodhiates (Eds.), *Technology in Education: Looking Towards 2020*. Hillsdale, NJ: Erlbaum.

Sternberg, R. J. (1984). *Beyond IQ: A triarchic theory of human intelligence*. New York: Cambridge University Press.

Super Factory. (n.d.). [Computer program]. Pleasantville, NY: Sunburst Communications.

The Factory. (n.d.). [Computer program]. Pleasantville, NY: Sunburst Communications.

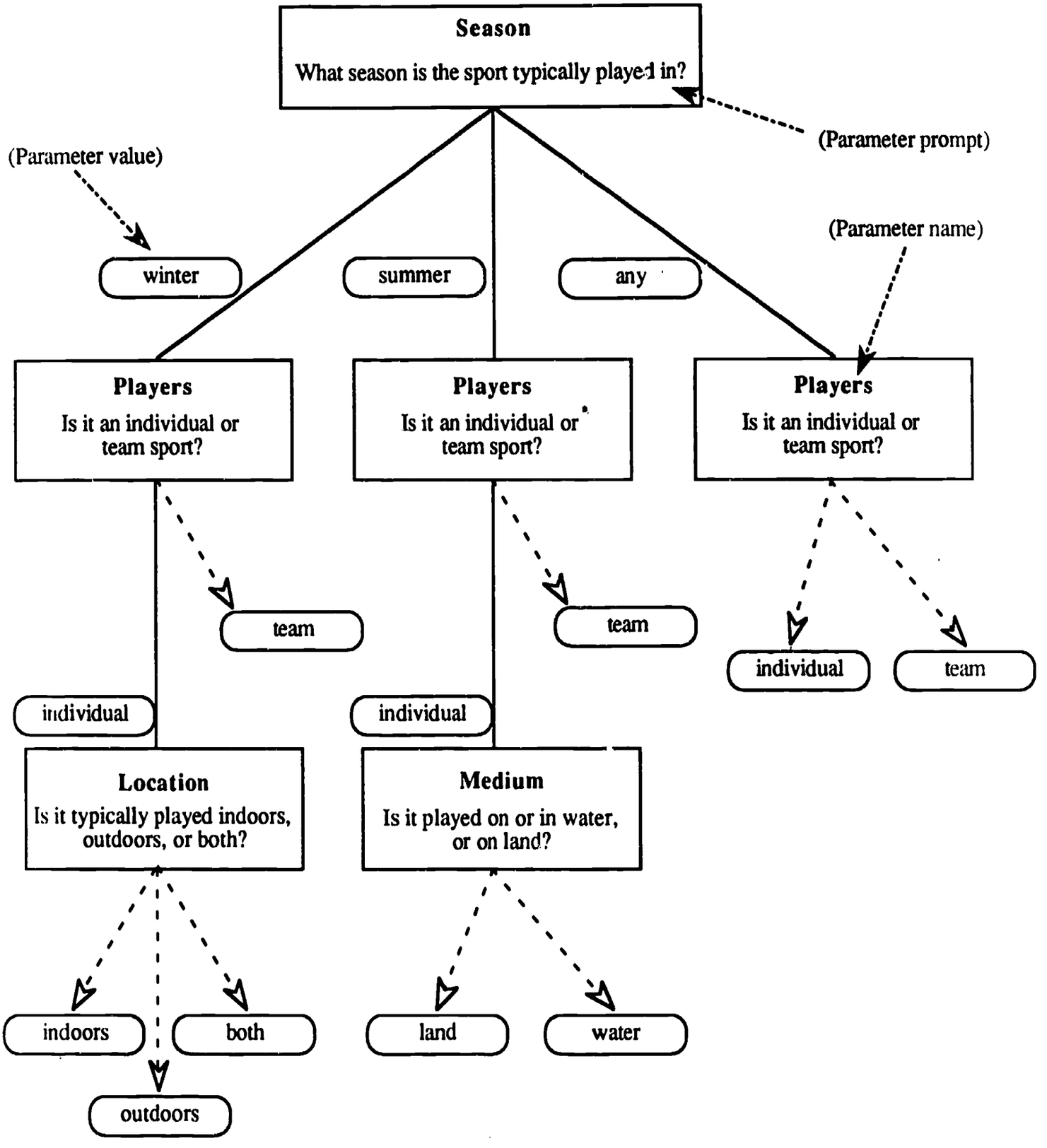
Thinking Skills Language Program (Vols. 1-9). (1978). Stamford, CT: Innovative Sciences.

Trollip, S. R., & Lippert, R. C. (1987). Constructing knowledge bases: A promising intructional tool. *Journal of Computer-Based Instruction, 14*, 44-88.

Vygotsky, L. S. (1962). *Thought and language*. Cambridge, MA: M.I.T. Press.

Wideman, H. H., & Owston, R. D. (1988). Student development of an expert system: A case study. *Journal of Computer-Based Instruction, 15*(3), 88-94.

Appendix A



Sample tree diagram with sports KB (broken arrows indicate chart branches not completely illustrated.) Note how the same parameter may be used in different rule branches.

Appendix B

Weather Project Assignment

Types of weather to be predicted by your expert system:

Rain

Thunderstorms

Clearing

Remaining fair

Temperature:

rising
falling

Cloud cover

Criteria to use in making prediction rules:

temperature
temperature change

barometric pressure
pressure change
rate of pressure change

wind direction
wind direction shift - backing/veering
wind speed and speed change

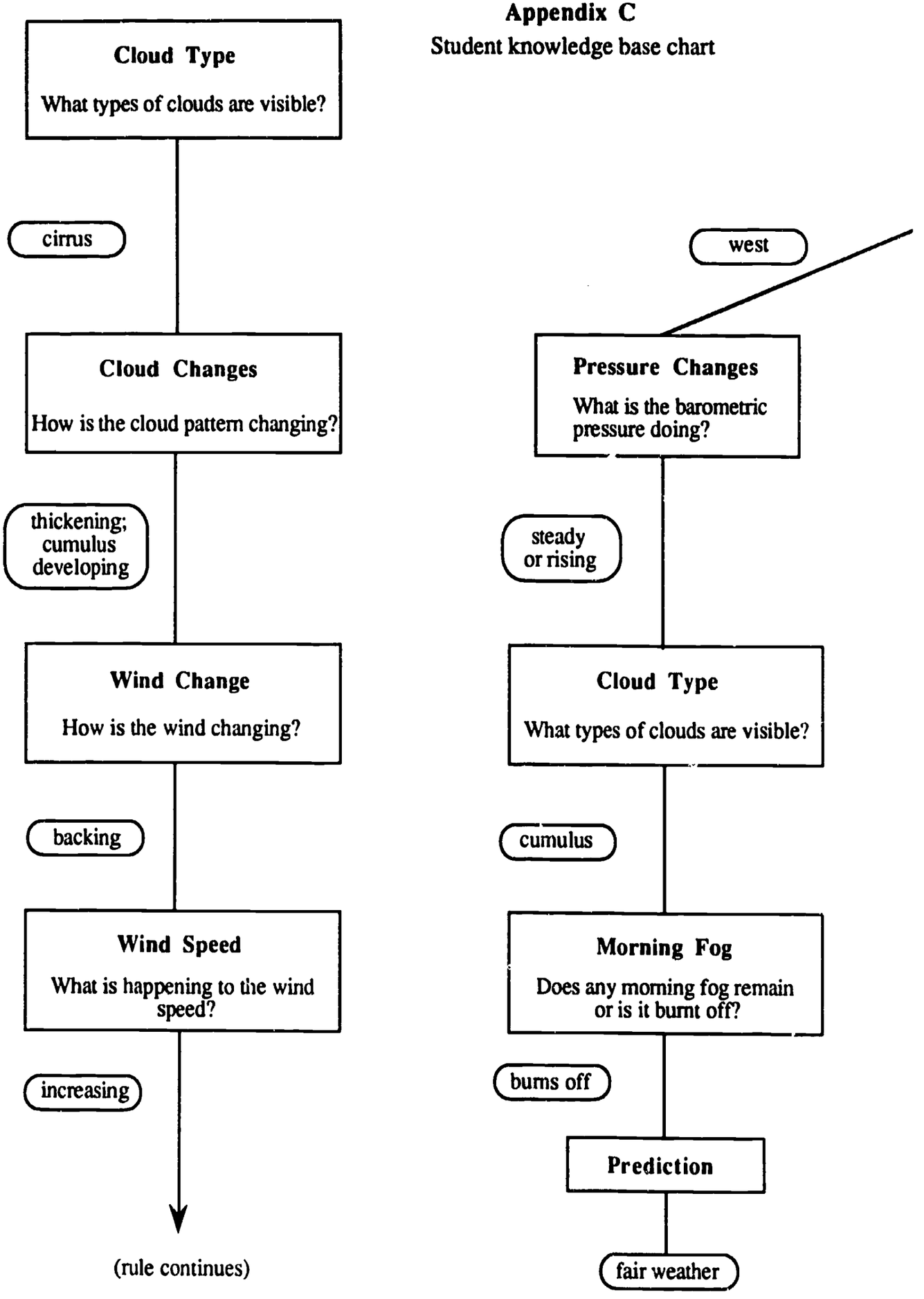
degree of cloud cover
cloud types
changes in cloud types
cloud speed

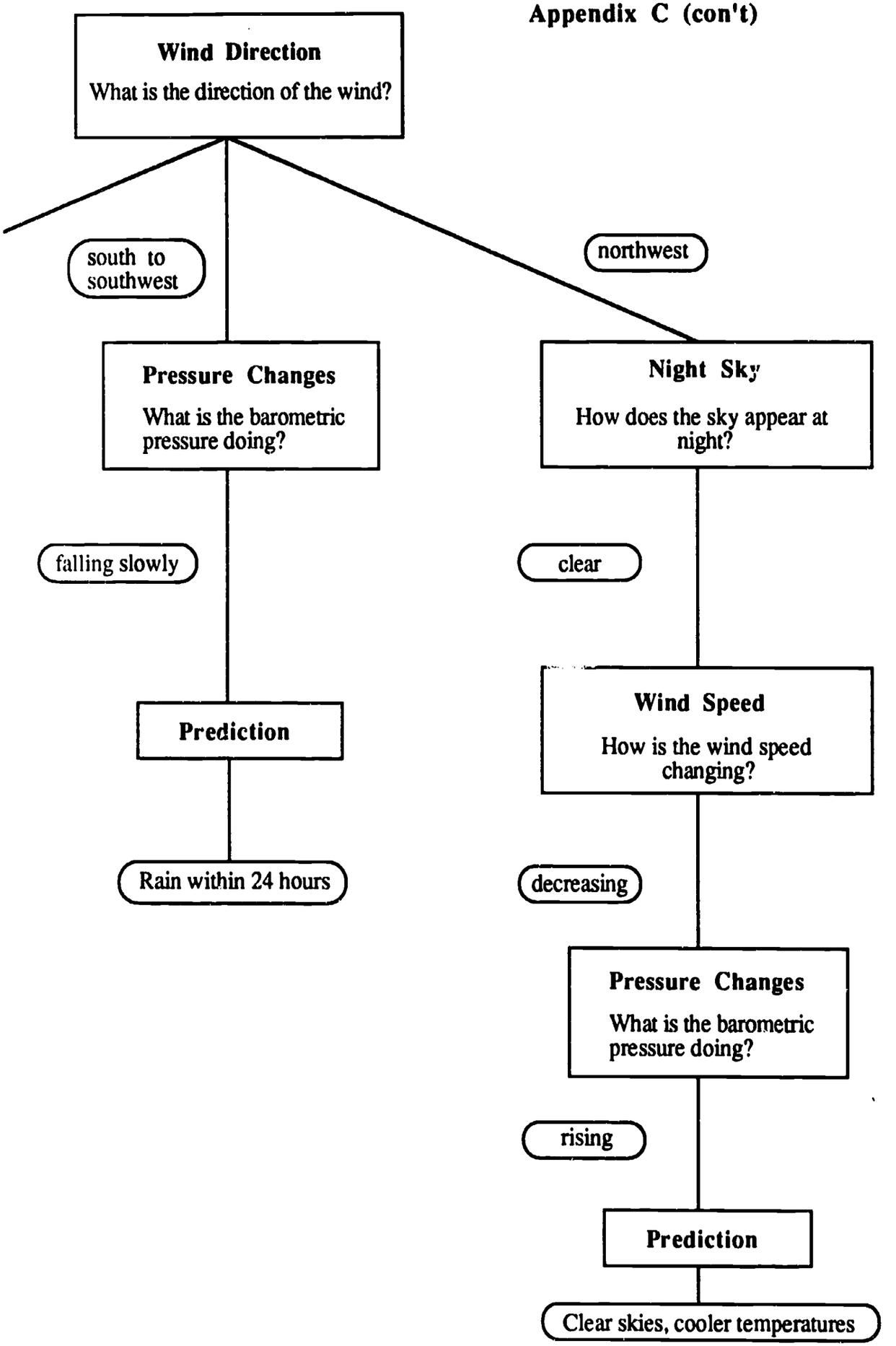
humidity

recent weather

Appendix C

Student knowledge base chart




Wind Direction
What is the direction of the wind?
south to southwest
northwest
Pressure Changes
What is the barometric pressure doing?
Night Sky
How does the sky appear at night?
falling slowly
clear
Prediction
Wind Speed
How is the wind speed changing?
Rain within 24 hours
decreasing
Pressure Changes
What is the barometric pressure doing?
rising
Prediction
Clear skies, cooler temperatures

The Centre for the Study of Computers in Education is dedicated to encouraging collaborative research between schools, industry, government, and the university. Inquiries may be directed to: Centre for the Study of Computers in Education, Faculty of Education, York University, 4700 Keele St., North York, ON, M3J 1P3