This is to certify that the

thesis entitled

REPRESENTATION AND PROCESSES

OF PEDAGOGIC KNOWLEDGE

presented by

HAROLD CHARLES GROSSMAN

has been accepted towards fulfillment
of the requirements for

Ph. D. degree in Computer Science

Richard J. Reid

Major professor

Date 5-20-78

O-7639

REPRESENTATION AND PROCESSES OF PEDAGOGIC KNOWLEDGE

By

HAROLD CHARLES GROSSMAN

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1978

ABSTRACT

REPRESENTATION AND PROCESSES

OF PEDAGOGIC KNOWLEDGE

By

Harold Charles Grossman

Computer-based, generative instructional systems have increased in occurrence as innovative methods of instruction are developed. While generative instructional systems are capable of the production of many problems, the systems cannot solve these problems. The knowledge representation and processes of these systems do not cover more than one subject area. This research develops a representation and processes for any pedagogic subject area that can be expressed as a set of transformations.

A pedagogic knowledge representation is defined for a task-oriented subject area. The thrust of the subject area is the application of algorithms to data structures. Three relations, simple transform, transform, and subset, are identified. A diagram calculus is developed to identify all transformations in a representation. A functional automaton, whose language is the set of all functional compositions in a representation, is derived from a representation.

Pedagogic path, dependency, and partial ordering of concepts are derived from a representation. Trivial, simple, complex, complete, and extrinsic problems are defined and discussed relative to a pedagogic

representation. A solution planning sequence which solves problems independent of the problem generation process is shown to compute the answer to the stated problem.

ACKNOWLEDGMENTS

The author wishes to expresses his particular appriciation to Richard J. Reid, his academic advisor and chairman of his doctoral committee, who generously gave his time and encouraging advice throughout the research that lead to this work. The author would also like to thank Martin Keeney, Carl Page, Lewis Greenberg, and Jacob Plotkin for their participation in the doctoral committee and their numerous suggestions for improving the research.

LIST OF FIGURES

FIGURE		PAGE
II.3.3	Digital Circuits Concept Tree	24
II.4.3	$P_i \cup P_j$	29
II.4.4	$P_i \cdot P_j$	29
II.4.5	$P_i \cdot/\cdot P_j$	29
III.2.12	Pedagogic, Functional Diagram	46
III.7.1	Pedagogic, Functional Diagram of "Angles"	71
III.7.2	Functional Automaton of "Angles" . . .	71

CHAPTER I

INTRODUCTION

The notion of a generative, computer-assisted, instructional system has attracted some attention. Carbonell(2), Vickers(35), Ramani and Newell(24), Brown and Burton(1), and Koffman and Perry(10), among others, have reported on such generative systems. The endeavor at hand is not to devise another specialized generative instructional system but to devise powerful and general mechanisms to represent a large class of such generative systems.

The current view in the field of artificial intelligence is that intelligence will result when information and processes of an appropriate form and content are constructed. The attempt at construction of such processes is to be complemented in artificial intelligence by studying their actual and potential structure, and the structure of the information that they incorporate or might incorporate. The research that is reported in this dissertation is the application of the field of artificial intelligence to the problem of a general representation of pedagogic knowledge in a large class of generative, computer-assisted, instructional systems.

I.1 Overview

A traditional computer-assisted, instructional system is a collection of computer programs that simulate one of the numerous activities associated with teaching. A generative, computer-assisted, instructional system is an instructional system that performs the instructional activities from some representation of the knowledge contained in the subject area. The exact form and processes of the information contained in these representations differ from system to system and from subject area to subject area. Most generative instructional systems use ad hoc techniques that are devoid of identifiable form or process. By studying those generative instructional systems that do contain some form or process, a general representation of the knowledge contained in these systems may emerge.

The generative instructional systems can be categorized into two classes via their design and usage. The first class contains the attribute question-answer systems. In these systems a question is asked pertinent to some attribute of a concept in the given subject area. The answer is typically one word or a short English phrase. The answer can also be computed from the representation of the knowledge in the subject area. Thus the attribute question-answer systems have sufficient knowledge and processes to ask meaningful attribute-type questions, relative to the subject area, and to answer similar questions posed to the system about the subject area.

The second class of generative systems is called problem generators. The problems generated by these systems require the application of an algorithm upon the data structures that are produced by the system. The answers to the generated problems are not determined by the knowledge representation that is incorporated in the system. The answers, if they are computed by the system, are computed by special purpose routines. These routines are associated with the generation of the problem and possess none of the typical problem solving abilities necessary for a solution that is independent of the problem generation process. Unlike the first class of generative instructional systems, the problem generators do not use a representation of the knowledge in a subject area to solve problems within that subject area.

The subject area of a problem generating system is a collection of facts and actions that are modelled by the activities which are simulated by the particular system. A task-oriented subject area is a subject area that is viewed from the data structures representing concepts within the subject area and the algorithms that manipulate those structures. The intrinsic elements of a task-oriented subject area are the data structures and the algorithmic actions that transform one data structure into another. The second class of generative instructional systems covers task-oriented subject areas.

I.2 Historical Vista

The usage of computers in assisting in the education of students is attracting some attention, as new and innovative, alternative approaches to the traditional models of instruction are being developed. This usage has developed in two major areas. The first area deals with teaching students new material about a given subject area. These instructional systems are the traditional computer aided instructional systems. The second major area is primarily concerned with the usage of computers to test students. This latter area has mainly concentrated on the construction of tests from a large data file of questions.

Recent inclusion of artificial intelligence techniques in specific generative instructional systems have yielded a superior, more natural interaction between the student and the particular system. This development has attracted some researchers in the artificial intelligence area of computer science and may yield some interesting results with the application of artificial intelligence techniques to the various aspects of constructing meaningful problems within a given task-oriented subject area.

The more traditional computer aided instructional systems, such as PLANIT, COURSEWRITER, and PLATO, use an ad hoc-frame-oriented approach as described by Carbonell(2). In this approach instructional material is stored in a data base that consists of many frames of specific pieces of text and questions. The frames include correct answers,

and frequently, anticipated wrong answers, keywords, and branching. The data base is entered in advance by the constructor of the instructional system. Essentially once the system is specified, the instruction is identical for each student. All of the traditional computer aided instructional systems lack the dynamics of a normal, instructional interaction between student and teacher.

To more closely model a typical teaching situation, Carbonell(2) designed and implemented a computer aided instructional system called SCHOLAR. Carbonell described SCHOLAR as an information-structure-oriented, computer aided instructional system, because SCHOLAR is based on the utilization of an information network of facts, concepts, and procedures. By using the knowledge as specified in an information-structure-oriented knowledge representation, SCHOLAR can generate text, questions, and corresponding answers. SCHOLAR can also utilize its information-structure-oriented network to solve questions formulated by the student. Thus a mixed initiative dialogue between the student and instructional system more appropriately simulates the student-teacher interaction.

The second category of computer-assisted instructional systems is a more recent development of using a computer to assist in the construction of tests. Several such systems have been in existence at Michigan State University since 1971. One widely known test construction program is the Prosser exam generator(22), designed and developed by F. Prosser of Indiana University. This exam generator

allows for the selection of questions from a data file either randomly or by instructor specified codes. The questions are printed on the left side of the printout page while the answers to those questions are printed on the right side of the page. The answers are computed before hand and stored along with the questions on a data file. Using this banking and retrieval mode of operation, the composition of a test or homework exercise can be automated along with answers to permit checking of student responses.

Vickers(35) proposed using a syntax-semantic approach for building questions which would involve the construction of valid and invalid FORTRAN variable names, expressions, and statements. The syntax-semantic approach, as described by Vickers, involved the writing of specific routines for each kind of question, i.e. one routine would construct valid, invalid, real, or integer variable names. The resulting system is expandable by writing new routines for each additional type of question to be built.

Izquierdo(8) and Whitlock(36) extended the original syntax-semantic ideas of Vickers to a general purpose routine that uses a set of productions to construct the desired question. This latter approach provides a flexible approach to the generation of questions and begins to identify some possible forms and processes necessary for the construction of a large class of questions.

Ramani and Newell(24) investigated the design and implementation of a problem generating system that constructs

programming assignments from an abstract representation of the problem. Design strategies utilizing grammar-like mechanisms to put together the compatible elements to create an acceptable problem structure are described. Attention is given to the mechanisms necessary for controlling the generation of coherent English cover stories; of problems involving specified concepts in a given subject area; and making use of information available about the concepts involved in the subject area. Other strategies discussed by Ramani and Newell are: generalized good problems to produce useful variants of problems; the use of reasoning about actions in a task area to recognize good problem situations; and the design of surface details to create problems having a given abstract structure.

Brown and Burton(1) describe a computer-assisted instructional system that is called SOPHIE. The problem situation in SOPHIE has a student trouble-shooting a particular piece of faulty electronic equipment, namely a Heathkit IP-28 regulated power supply. The student can ask for various measurements, make hypothetical changes, and ask for a list of possible faulty electronic components that is consistent with prior measurements. A backward working specialist and a forward working deduction system are used to develop a list of faulty components. SOPHIE also incorporates a semantically oriented predictive parser to carry out a natural language dialog with the student.

Koffman and Perry(10) report on an intelligent computer

aided instructional monitor and generative tutor that they have developed at the University of Connecticut. The tutor has a simplified English interface with the student and produces problems from some form of representation of the subject area. The exact form of the knowledge representation is not clearly specified. The knowledge may be directly programmed in the routines that generate and solve the problems in the system. The solutions of the generated problems are computed by specific LISP routines. The LISP routines do not have the ability to solve a problem that is not generated by the tutor.

Earl Sacerdoti(28) at Stanford Research Institute is developing a methodology for an organization of plans of actions for a computer based consultant. The computer based consultant is an expert on the assembly and disassembly of an air compressor. The plans for working on an air compressor are built in a representation called a procedural net. Although immediate application of the ideas and techniques being developed by Sacerdoti to the solution of problems in a task-oriented subject area are not clear, the class of activities of formulating plans and carrying those plans to their conclusion is in the category of the kinds of activities that an answer formulator for a problem generating system would have to possess.

Several areas of artificial intelligence research are tangential to the development of a knowledge representation that is suitable for a large class of problem generating systems. Novak(18,19) has developed a computer program,

ISAAC, which solves physics problems stated in English. The knowledge representation that Novak uses is a canonical object frame which is similar to a semantic network form of representation. de Kleer(5) describes a program, NEWTON, that employs multiple strategies in the solution of classical mechanics problems. Sussman(32) describes a problem solver for electrical design. The solution technique is stated in terms of "problem solving by debugging almost-right plans." Sussman's belief is that the creation and removal of bugs is an unavoidable part of the process of solving a complex problem.

The application of artificial intelligence techniques to instructional systems of the attribute question-answer category enhances the performance and interaction of these systems. A more natural student-teacher interaction that closely models the typical instructional environment is the direct result of the inclusion of artificial intelligence techniques. Such systems appear to be superior to their non-artificial intelligence counterparts.

I.3 Context of Research

The goals of artificial intelligence researchers are diverse, so it is necessary before going too far to understand how the present work is related to artificial intelligence's major subareas. Nilsson(16) has divided the field of artificial intelligence into a number of research areas, of which four are designated core areas, and the rest, first-level applications areas. The four

core areas are: common-sense reasoning, deduction, and problem solving; modeling and representation of knowledge; heuristic search; and artificial intelligence systems and languages. The present effort is in the second category but also touches on the first category. The principal approach of research in those areas is the building of understanding systems that embody knowledge about some subject area, that are able to manipulate that knowledge, including problem solving, and that are able to exhibit communicative behavior to demonstrate their abilities and the content of knowledge that they contain.

A number of past efforts of constructing generative instructional systems have dealt with various aspects of some of the problems encountered in building understanding systems(1,2,8,9,10,18,19,24,27,35,36). None have dealt with the form of the representation of the knowledge content for more than one subject area. Only two generative instructional systems contain the ability to solve their own questions independent of the generation process(1,2).

Several lines of research that are relevant to various components of generative instructional understanding systems can be mentioned: problem-solving techniques and solving simple puzzles using means-ends analysis and heuristic search; the representation and subsequent use of structured knowledge in semantic networks; and the use of ad hoc problem solving procedures along with the use of predicate calculus notation and general uniform proof procedures.

In addition to the necessary scope limitations of

many of the results of research in the past, there are some broader respects in which the research is inadequate for attacking the larger aim of building knowledge representations for generative instructional understanding systems. The effectiveness of any single representation over a diverse set of subject areas has not been demonstrated. To aim at such a representation is desirable at least from the standpoint of parsimony, though parsimony might turn out to be unachievable. In dealing with knowledge representations, each system represents its own task domain without attempting to address any of the typical examples of the other systems, or to deal with specific problems (representational and processing) in the other approaches. This results in a collection of systems covering a number of task areas but whose interactions and overlaps are quite unknown. As a result it is difficult to determine if particular research is a real advance.

Very few systems have a coherent approach to one of the primary problems of the area, the knowledge interaction problem. SCHOLAR and SOPHIE are examples of systems that do address the knowledge interaction problem. This problem can be partially approached by asking how knowledge is applied when appropriate, how its appropriateness is recognized so that it can be brought to bear, and how it interacts with other pieces of knowledge in ensuring a correct result when single pieces of knowledge or single knowlege sources are insufficient by themselves.

There are a number of essential properties, from a

conceptual standpoint, of a knowledge representation if it is to be used effectively for a generative instructional understanding system. Moore and Newell(12) give a list of dimensions on which understanding systems are to be evaluated. This list includes representation, assimilation, efficiency, accommodation, action, directionality, depth of understanding, and error. Without elaborating on the definitions of these dimensions, it can be seen that these are high-level properties of a system. Rather than using those traits directly, Rychener(27) suggests that it is more useful to focus on the representation, and see how the various traits imply desirable properties for the system. The following list of properties of knowledge representations has emerged, though it is not to be put into direct correspondence with the list of eight dimensions suggested by Moore and Newell.

Knowledge within the structure should have the following characteristics:

1. Encodability - knowledge should be easily mapped from an external form to the form in the understanding system; ultimately, the encoding should be accomplished via an automaton.

2. Inspectability - content of knowledge already encoded should be readily derivable; this is the converse of encodability, and perhaps could also be called extractability.

3. Accessibility - knowledge should be accessible in some form for application when it is appropriate; this

need not be as complete an operation as for inspection; when knowledge is accessed or applied, its own nature is not evident as is its effect.

4. Operability - knowledge must be amenable to such operations as mapping, forming analogies, generalizing, optimizing, reformulating, deducing, and inducing.

5. Flexibility - knowledge should have a number of alternative forms, for instance along the procedural-declarative aspect.

6. Organizability - there should be a variety of potential control organizations, according to the demands of various kinds of content knowledge.

7. Provability - there should be a way to guarantee correctness or perhaps consistency of the encoding, in some informal sense; this may include being able to justify the presence of some knowledge by knowing how it has been found necessary for some behavior.

I.4 Goals

The main goal of this dissertation is to establish a knowledge representation and processes for a large class of generative instructional systems in a task-oriented subject area. Just as all artificial intelligence research projects, by necessity, have been restrictive, the current effort is restricted to representations and processes for a pedagogic subject area.

The subject area is viewed from the concepts that compose the subject area. The concepts are represented by

data structures and algorithms. The knowledge that will be represented is algorithms and data structures that are the concepts in a pedagogic subject area. In the remainder of this work we will call a formal task-oriented subject area, a subject area, and by that mean a pedagogic subject area composed of concepts that are represented by algorithms and data structures.

A direct benefit of developing a general representation of the pedagogic knowledge content in a subject area is that all generative instructional systems could use the same representation schema. The constant schema would permit experimental comparisons of different systems possible with the measurement of such properties as smartness or flexibility. Other benefits would be greater emphasis of the processes that are involved in generative instructional systems and a basis for departure of future work in this area.

The processes that complement the representation in a generative understanding system, as described by Ramani and Newell(24), can be grouped into three areas. The representation and processes must be sufficiently rich to 1) answer pedagogic questions about an encoded subject area, 2) construct complete, meaningful problems from an encoded subject area, and 3) construct a sequence of problem solving steps from which an answer to any possibly generated problem from a given representation could be computed.

The first capability provides the means for extraction of the knowledge content within a given representation.

Without this ability the constructor of an encoded subject area would be unable to justify to himself, herself, or others that the encodement was complete with respect to any criteria. The first capability also permits a measurement of pedagogic aspects of a specific representation.

The more interesting capabilities are the second and third ones. These abilities provide a basis from which to address questions such as: what is a good, meaningful, complete problem; when can a problem be solved; can a paraphrase of another problem be identified; does the construction of problems from a given representation have any analogy to the human process of construction of problems; and do the problem solving steps have any analogy to the human process of problem solving. The major thrust of this work is the development of a theoretical foundation to address the above questions.

I.5 Organization

The remainder of the dissertation is organized into four chapters. Chapter II is a study of various knowledge representations that have been used in artificial intelligence research and in generative instructional systems. Representations including syntax-semantic-like forms, semantic networks, and concept trees are viewed relative to the goals outlined in chapter I.

Chapter III is the development of a knowledge representation in terms of a set of primitive relations and the composition of those relations. Mathematical-like

properties of the primitive relations and their combinations are included. Several subject areas are encoded into the developed representation.

Chapter IV presents pedagogic implications that are derivable from a representation. A problem is defined relative to the developed representation and a meta-grammar is presented to determine complete problems. A problem solution sequence is shown to solve complete problems in a given representation. Chatper V concludes the work and suggests some extensions to the present research.

CHAPTER II

LITERATURE REVIEW

Several alternative representations have been used to represent the knowledge content in a computer-assisted instructional system. Grammar-like mechanisms have been the most prolific with respect to systems whose main thrust is the generation of problems(8,10,24,33,35,36). Semantic networks have been used in several instructional systems as a knowledge representation(1,2,24). Concept trees have played a central role in several instructional systems as a means for describing a pedagogic ordering of concepts within a subject area(9,10). None of these mechanisms have been used for both the construction of problems and the solution of problems in a subject area.

II.1 Grammatical Representations

The simplest generative representation for problems is an extension of the "slot-and-filler scheme" as it is called in linguistics. In this scheme a problem is expressed as a set of problem frames incorporating variables. Each problem frame has a specification of the admissible values that the variables or slots of the frame may assume. The specification for the slots are either in the form of numerical limits or a set of admissible values for each

variable.

The slot-and-filler scheme can be enriched in two directions. Uhr(33) reports one direction which is to implement computationally defined relations between the variables of a problem frame. By choosing certain independent variables randomly, either within the permitted range of values or from sets of admissible values, the remaining variables could be computed from these randomly chosen variables. The computational relations would ensure that a meaningful problem is constructed from a compatible selection of values for the variables in the problem frame.

The second direction of enriching the slot-and-filler scheme involves the use of a context-free grammar. Using a context-free grammar, the problem frames are the language of the grammar. Thus unlike the slot-and-filler schemes using the problem frames described above, the grammar-like schemes are not limited to regular languages. The grammar-like schemes have been used in the context of problem generation for synthesizing data structures having phrase structure.

Simple programs that use grammar-like schemes to generate questions in artificial intelligence (developed by Newell and Robertson) and in cognitive psychology (developed by Waterman) have been in use at Carnegie-Mellon University since 1971(24). The generations produced by these programs are always grammatically and semantically correct with respect to natural language theory. Most but

not all questions generated are meaningful within themselves. The grammar-like generation schemes could also include probability assignments that allow certain choices of syntactic and semantic structures to be more or less frequent than other choices. This allows some control on the generated structure so that favorite problem structures are more likely.

Enumerative schemes, such as the above, derive their power from a basic device that is the classification of a set of phrases into different subsets of syntactically and semantically similar items. The problem frames and context-free grammar schemes are not sufficiently rich to express the knowledge content of a subject area to both generate and solve problems from the same representation.

Implicit representation of knowledge in grammar-like systems provides only two devices to ensure coherence of the generated problem: co-occurrence of symbols in the grammar rewrite rules which ensures co-occurrence of complementary parts of the problem; and set membership which enables semantically and syntactically equivalent items to be distributed in the problem in an identical manner. An inherent limitation with grammar-like systems is the absence of an explicit representation of the knowledge content of the kind found in practically every artificial intelligence research project.

II.2 Semantic Networks

Carbonell(2) chose to use semantic networks as the knowledge representation in his instructional system SCHOLAR. Carbonell developed SCHOLAR's representation from earlier work that Quillian(23) first reported in 1966. Quillian used semantic networks to model human memory and natural language comprehension. Carbonell extended Quillian's work to develop a knowledge representation that would pose questions to a student, check the student's answer, and most importantly answer student posed questions.

The particular dialect of semantic networks that Carbonell used is described as an organization of units of information in terms of their natural language meanings and relationships. Each unit of information may refer to other units within the set of information units. Each unit may also refer to other units which refer to other units or itself. Carbonell called this particular semantic network, a "semantic information network."

The units of information are nodes in the computer implementation of the knowledge representation. There are two kinds of nodes which are labeled type nodes and token nodes. In Carbonell's usage of semantic networks, a <u>type node</u> is a unit pointing to an informational, multi-level list of pointers to other units. Words referring to other nodes in the body of the unit are <u>token nodes</u>. Each token node represents a pointer to the corresponding type node, i.e. the unit with that word as a name. By using type

and token nodes, information is not necessarily duplicated, since it is stored only once at the type node. This type of knowledge representation is recursive and leads to circularities which do not represent a serious difficulty and are not necessarily undesirable.

Each unit may be thought of as pieces of information to which is associated a name. There is no one-to-one correspondence between units and names. Each unit is composed of three elements. The first element of each property is the name of the property or attribute. The second is a set of tags used by the executive routine. The third element is the value of the attribute. A value can be either a set of attributes of a pointer to a unit or a set of units modified by other attributes.

By encoding a subject area such as the geography of South America in the above semantic network, answering and asking questions about the subject area involves following pointers and type nodes until the appropriate token node is found. Thus all of the properties of a given subject area are abstracted to generalized attributes. For instance, some of the attributes relative to geography are population, size, cities in a country, countries in a continent, language of a country, and principle industries in a country. To answer questions such as "Name a country in South America?" or "What language is spoken in Argentina?," the appropriate attribute link (countries in a continent, language of a country, respectively) from one unit (South America, Argentina, respectively) to the answer

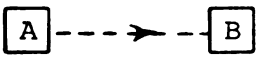
(Argentina, Spanish, respectively) is followed. This category of question and answer producer is very analogous to the question and answering systems developed for natural language processing of which some of the most notable are those by Schank(29) and Woods(39).

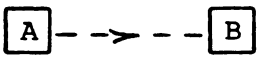
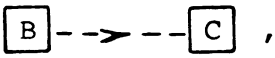
An internal representation of the knowledge in a subject area provides the flexibility to generate questions and answer questions about a subject area. The semantic network with its value-attribute-value representation allows for the generation or answering of attribute types of questions. For attribute types of questioning and answering, the semantic information network as a knowledge representation is a quite adequate structure.

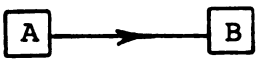
II.3 Concept Tree

Koffman(9) develops the idea of a concept tree to specify the order of presentation of course concepts via a generative instruction system. The nodes of a concept tree represent the course concepts and the branches of the tree represent one of two relations that are defined later. The concept tree represents the constructor's interpretation of the relationship between course concepts. Obviously, concept trees are not unique for a given subject area, just as different people interpret course concepts differently. The plateau upon which a given concept is placed indicates that concept's relative degree of complexity within the subject area. The concept tree for a digital circuits course, as developed by Koffman, is shown in Figure II.3.3.

The relations between the nodes of the tree can be defined as follows:

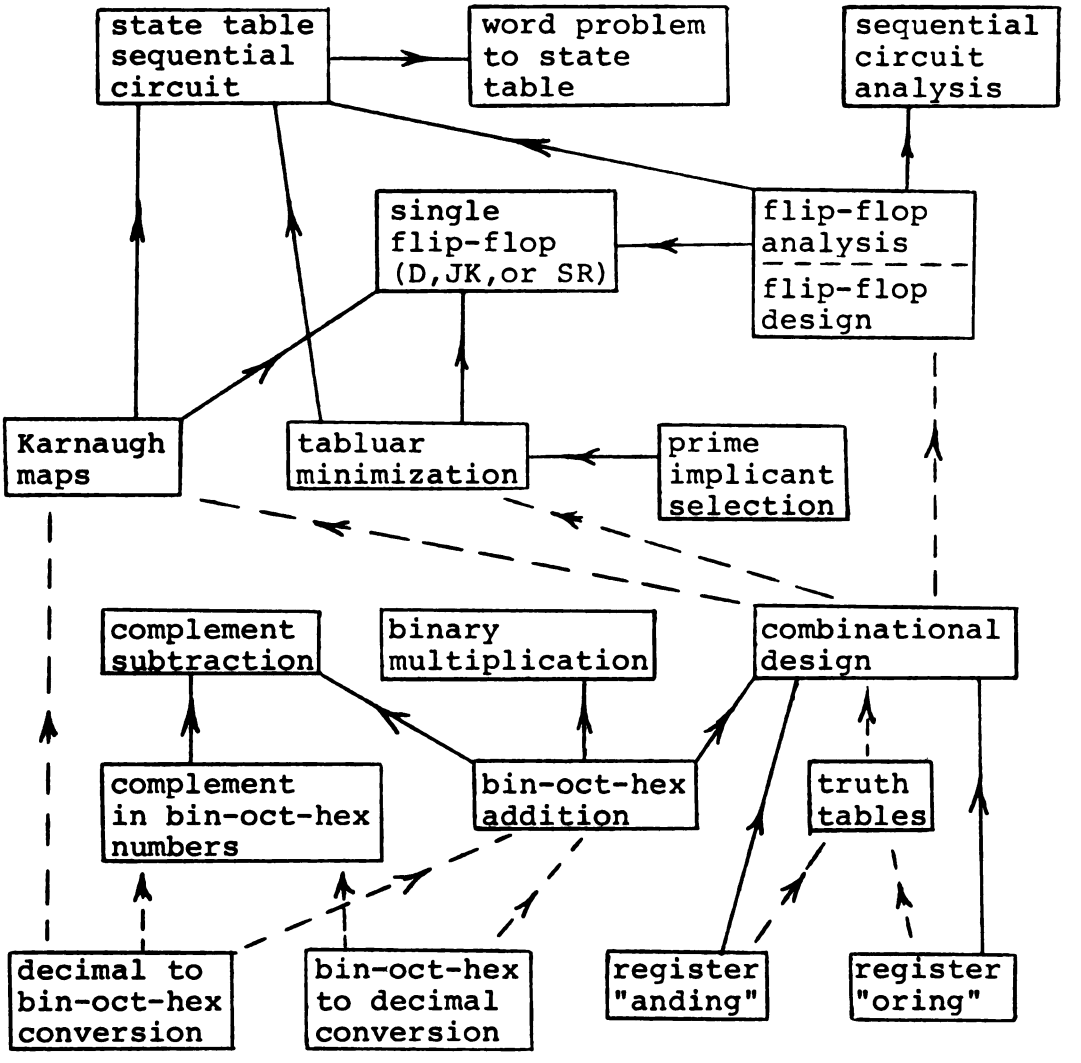
Definition II.3.1 Concept A is a prerequisite of concept B, denoted as  , means that concept A should be mastered before concept B.

The normal order of presentation and learning of a given concept would be all of the prerequisite concepts followed by the given concept. Notice that the relation "is a prerequisite of" is a transitive relation such that if  and  , then  .

Definition II.3.2 Concept A is a prerequisite of concept B and concept A may be used as a subconcept by B, denoted as  , means that concept A should be mastered before concept B and that when solving a problem involving concept B, solution techniques involving concept A might be used as a series of sub-tasks.

While Koffman does not mention any relational properties, the latter relation appears to be transitive such that if  and  , then  .

By using the above relations, Koffman constructs a concept tree for a given subject area. His instructional system uses the constructed tree to select appropriate concepts for the student to learn. In general, all concepts on a given plateau are mastered before advancing



Digital Circuits Concept Tree

Figure II.3.3

to the next higher level plateau. The second relation between course concepts begins to identify the importance of problem solving by using a knowledge representation of the subject area. Koffman does not use the relation in solving problems that his instructional system generates. Obviously, the concept tree in Figure II.3.3 is not a tree in the mathematical sense

II.4 Abstract Problems

Koffman and Perry(10) introduce the idea of a basic abstract problem as a structure representing certain subclasses of a given collection of problems.

Definition II.4.1 An abstract problem is a triple of the form ((unknown), (data), (relation)) where (unknown) is a list of variables whose values are sought as the solution to a problem represented, (data) is a list of input variables, and (relation) defines a function which assigns unique values to the unknowns for each list of data values.

A basic abstract problem is assumed to be solvable. A complex abstract problem is one that is constructed from several basic abstract problems. Each basic abstract problem will have a corresponding problem solver or solution method. The problem solver is determined by the problem generator. A complex abstract problem will be solvable by using a combination of the problem solvers associated with its basic abstract problem constituents.

Definition II.4.2 A problem generation and solving system is described by a triple (C, R, S) where R is a representation for the class of problems C which can be solved by the solution method S.

Koffman and Perry(10) indicate that the power of such a system can be measured by the size and variety of C. The power can be increased by 1) enlarging C while keeping S fixed, 2) enlarging C while extending S, or 3) decreasing the degree of specialization required of the user to implement S while keeping C and S (the method) fixed.

As an example, let (C_i, R_i, S_i) be a collection where C_i is a subclass of problems represented by R_i and solvable by S_i. The subclass of problems, C_i, deals mainly with a particular concept in a course. R_i could be a generative grammar or a generation routine which generates problems dealing with C_i. S_i solves the generated problems and monitors the student's solution. A generative instructional system can be extended by adding more basic elements. By defining relations on the basic elements, a structure such as a concept graph similar to Koffman's(9) concept tree can be identified.

The predicate of an abstract problem ((unknown), (data), (function)) is the predicate PR such that PR((unknown), (data)) is true if and only if the value(s) of the (unknown) are the value(s) given by the function defined by (function) when evaluated on (data). Predicates can be used for formal proofs concerning abstract problems. If the predicate of

a problem can be proven from the predicates of basic abstract problems, the structure of the solution in terms of basic solution procedures is determined by the proof.

The constructions on abstract problems involve set theoretic and functional operations via Koffman and Perry's(10) schema. Constructions include subproblem, specialization, generalization, analogy, transformation, union (or concatenation), composition, cascade, and domain dependent constructions. The constructions are implemented as operators which apply on the one hand to sublists R_i and on the other hand to solution routines S_i. The main syntactic generative operations are union, composition, cascade, and certain kinds of transformations. Subproblem is a decomposition techniques which yields, ultimately, basic problems. Specialization, generalization, and analogy are special cases of transformation.

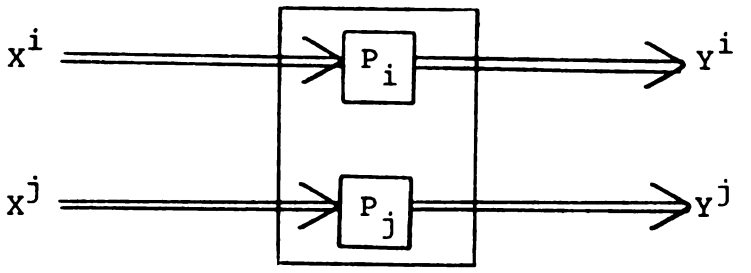
Let P_i and P_j be two abstract problems with $P_i = ((Y^i), (X^i), (S_i))$ and $P_j = ((Y^j), (X^j), (S_j))$ with predicates $PR_i = ((Y^i), (X^i))$ and $PR_j = ((Y^j), (X^j))$, respectively. The union or concatentaion of P_i and P_j, denoted $P_i \cup P_j$, is defined to be $(((Y^i) \cup (Y^j)), ((X^i) \cup (X^j)), (S_{P_i \cup P_j}))$ where $S_{P_i \cup P_j}$ is a solution routine for all the problems represented. $S_{P_i \cup P_j}$ is determined by the conjunction of PR_i and PR_j, i.e. $PR_i((Y^i), (X^i)) \wedge PR_j((Y^j), (X^j))$. "U", union, on the variables indicates

set union. The superscript denotes a list of variables. Diagrammatically, the union of P_i and P_j is shown in Figure II.4.3.

Let P_i and P_j be the same as before except that Y^i is a single element, Y_i, and $(Y_i) \subseteq (X^j)$. $P_j \cdot P_i$, the composition of P_i and P_j, is defined to be $((Y^j), ((X^i) \cup ((X^j)-(Y_i))), (S_{P_j \cdot P_i}))$ where $S_{P_j \cdot P_i}$ is determined by the predicate $PR_i((Y_i), (X^i)) \wedge PR_j((Y^j), ((Y_i) \cup ((X^j)-(Y_i))))$. Diagrammatically, $P_j \cdot P_i$ is given in Figure II.4.4.

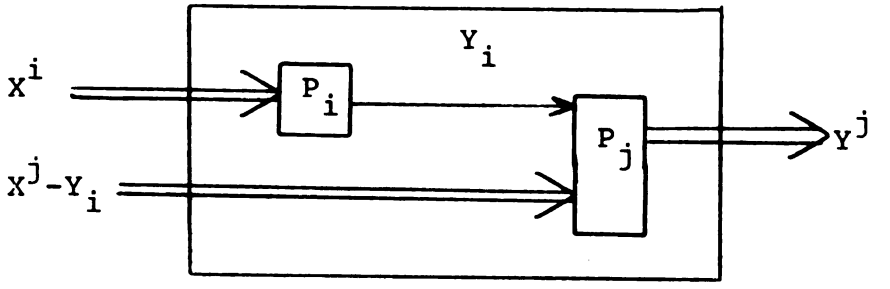
The cascade operation is a generalization of both the union and composition operations. Let $P_i = ((Y^i), (X^i), (S_i))$ and $P_j = ((Y^j), (X^j), (S_j))$ and also assume that $(Y^i) \cap (X^j)$ is not empty. $P_j \cdot/\cdot P_i$, the cascade of P_i and P_j, is defined to be $(((Y^j) \cup ((Y^i)-((Y^i) \cap (X^j)))), ((X^i) \cup ((X^j)-((Y^i) \cap (X^j)))), (S_{P_j \cdot/\cdot P_i}))$ where $S_{P_j \cdot/\cdot P_i}$ is determined by the predicate $PR_i((Y^i), (X^i)) \wedge PR_j((Y^j), (((Y^i) \cap (X^j)) \cup ((X^j)-((Y^i) \cap (X^j)))))$. The cascade of P_i and P_j is illustrated in Figure II.4.5.

Another natural relation which can hold among abstract problems is that of a map or transformation. Let P_i and P_j be defined as before. A map from P_i to P_j is a pair (f_1, f_2) where $f_1: (Y^i) \longrightarrow (Y^j)$ and $f_2: (X^i) \longrightarrow (X^j)$



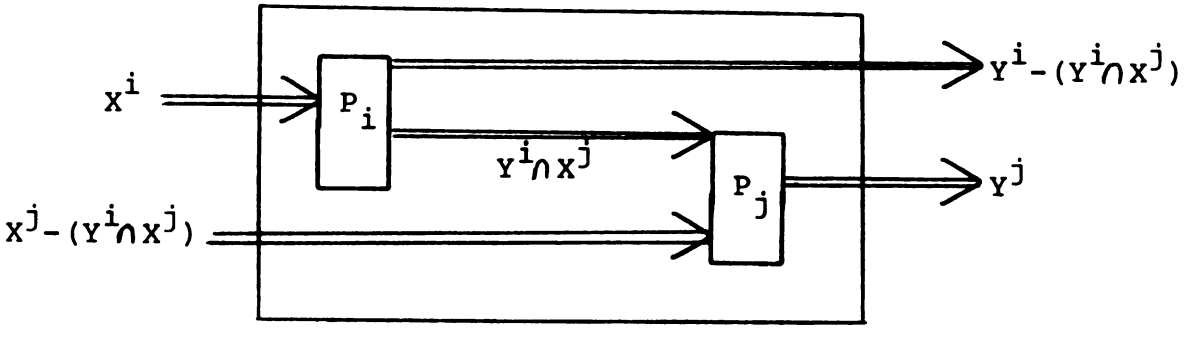
$P_i \cup P_j$

Figure II.4.3



$P_i \cdot P_j$

Figure II.4.4



$P_i \cdot/\cdot P_j$

Figure II.4.5

such that if $PR_i((Y^i), (X^i))$ is true then $PR_j(f_1((Y^i)), f_2((X^i)))$ is also true. If P_i is an abstract problem and $f = (f_1, f_2)$ is a pair of functions such that $PR_i((Y^i), (X^i))$ is true implies that $PR_i(f_1((Y^i)), f_2((X^i)))$ is true, then the construction by map gives the abstract problem $f(P_i) = ((f_1((Y^i))), f_2((X^i))), (S_i))$. If the condition on the predicate does not hold, it will be necessary to modify S_i to obtain a new solution routine.

The shortcomings of semantic networks is that the attribute in the value-attribute-value link is difficult to generalize to any task-oriented subject area. The necessary attributes have been identified for the specific subject area "mechanics"(5,18,19). However, the generalization of these or other attributes to any task-oriented subject area seems difficult.

The syntax-predicate representation that is developed by Koffman and Perry(10) has several shortcomings. The representation does not allow the discovery of additional knowledge from the original encoded subject area. The solution of a problem does not solve a problem but just provides an answer in conjunction with the generation of a problem. The solution procedure of complex problems, i.e. S_{P_j} ./. P_i, is not shown to solve the stated problem, is not shown what it is, and is not shown how to compute it. The manipulation of knowledge and the processes of the syntax-predicate representation are shortcomings with Koffman and Perry's representation.

CHAPTER III

REPRESENTATION OF PEDAGOGIC KNOWLEDGE

A representation of pedagogic knowledge is presented for subject areas that are composed of declarative and procedural concepts. Three binary relations, "simple transform," "transform," and "subset," are used in the description of a knowledge representation. A diagram calculus is developed to construct pedagogic, functional diagrams from a graphical representation of the simple transform and subset relations. Elements of the transform relation are discovered by manipulations of the diagrams.

A functional automaton is developed which accepts all functional compositions of a pedagogic knowledge representation. Non-repetitive functional compositions are computed from a matrix-type calculation. The chapter is concluded with an example subject area "angles."

III.1 Relations

Definition III.1.1 A <u>subject area</u> is identified by a finite non-empty set C of concepts. C = D U P where D is a non-empty set of declarative concepts and P is a non-empty set of basic procedural concepts. A <u>declarative concept</u> is a set data structure. A <u>procedural concept</u> p_k is an (n, m) recursive function where the domain of p_k

consists of n-tuples and the range of p_k consists of m-tuples.

Notationally, an element of a set is denoted by a small letter and a subscript, e.g. an element of P is p_k.

Examples of declarative concepts are a set of regular grammars, a set of directed graphs, and a set of Turing machines. A set will be named by the objects in that set, e.g. a set of regular grammars will be called "regular grammar." Examples of procedural concepts are the mapping of a context-free grammar into a grounded context-free grammar, the mapping of a graph into a path between two nodes in the graph, and the mapping of a push-down automata into the language of the push-down automata. Each element of P will be denoted by a name that represents the mapping, e.g. the mapping of a context-free grammar into a grounded context-free grammar will be called "grounding."

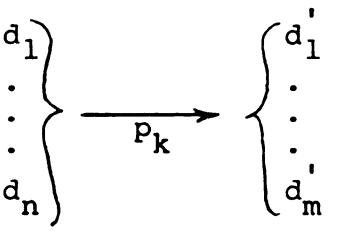
The procedural concepts are more important than the declarative concepts. The center of activity of this research is the procedural concepts or algorithms that determine the problems which can be formulated in C.

Three relations are identified with respect to C.

Definition III.1.2 A functional t from P to binary relations, called <u>simple transform</u>, is defined as follows:

$$t(p_k) = \{(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d_j') \mid d_i, d_j' \in D \text{ for } 1 \leq i \leq n \text{ and } 1 \leq j \leq m, \text{ and } p_k: \prod_{i=1}^{n} d_i \longrightarrow \prod_{j=1}^{m} d_j', \text{ and } \prod_{i=1}^{n} d_i \subseteq \text{domain } p_k, \text{ and } \prod_{j=1}^{m} d_j' \subseteq \text{range } p_k\}$$

Graphically, an element of $t(p_k)$ is denoted by



.

The brace may be omitted if the domain or range of p_k is a single declarative concept.

The relation t describes all basic functional relationships between the procedural and declarative concepts in C. An implicit assumption is that the complete domain and range of p_k is not necessarily known. All that is known about p_k is at least one subset of the domain and one subset of the range. The relation $t(p_k)$ does not necessarily contain all pairs of elements in C.

Example III.1.3 Consider the transformation called "directly produces" which maps a "sentential form" of a grammar into a "sentential form" of that grammar. The concepts in the example are

C = {directly produces(DP), sentential form(SF)}

where the abbreviations of the concepts are enclosed in parenthesis. The procedural concept is

P = {directly produces(DP)},

and the declarative concept is

D = {sentential form(SF)}.

The transformation is

DP: SF ⟶ SF .

Then by Definition III.1.2

t(DP) = {(SF, SF)}.

The graphical representation is

$$SF \xrightarrow[DP]{} SF \quad .$$

The above example illustrates a single declarative concept in the domain and range of a procedure.

Example III.1.4 A second example involves the declarative concepts

D = {syntax tree(ST), grammar(G), sentential form(SF), left-most canonical derivation(LMCD)}

and the procedural concepts

P = {structuring(S), derivation sequences(DS)}.

The transformations are

S: $ST \rightarrow G \times SF$

and

DS: $ST \times G \rightarrow LMCD$.

Then by Definition III.1.2

$t(S) = \{(ST, G \times SF)\}$

and

$t(DS) = (ST \times G, LMCD)$.

The graphical representation is

$$ST \xrightarrow[S]{} \begin{Bmatrix} G \\ SF \end{Bmatrix} \quad \text{and} \quad \begin{Bmatrix} ST \\ G \end{Bmatrix} \xrightarrow[DS]{} LMCD \quad .$$

The example depicts multiple declarative concepts in the domain and range of procedural concepts.

Example III.1.5 Another example involves the declarative concepts

D = {left linear grammar(LLG), right linear grammar(RLG),

language(L), sentential form(SF), finite state acceptor(FSA)}

and the procedural concept

P = {grammar intersecting(GI)}.

The mapping is

GI: LLG × RLG ⟶ L × SF × FSA .

Then

t(GI) = {(LLG × RLG, L × SF × FSA)}.

The element is represented by

$$\left.\begin{matrix}\text{LLG}\\ \text{RLG}\end{matrix}\right\} \xrightarrow{\text{GI}} \left\{\begin{matrix}\text{L}\\ \text{SF}\\ \text{FSA}\end{matrix}\right. .$$

The above example displays multiple declarative concepts in the range and domain of a procedural concept.

Definition III.1.6 A binary relation S on D, called <u>subset</u>, is defined as follows:

$S = \{(d_i, d_j) \mid d_i \subset d_j \text{ where } d_i, d_j \in D\}$.

Graphically, an element of S is represented by

.

Subset, as usual, is not reflexive or symmetric but is transitive.

Example III.1.7 Let the declarative concepts in C be

D = {phrase structure grammar(PSG), context-sensitive grammar(CSG), context-free grammar(CFG), linear

grammar(LG), regular grammar(RG)}.

Then

S = {(CSG, PSG), (CFG, PSG), (LG, PSG), (RG, PSG),
(CFG, CSG), (LG, CSG), (RG, CSG), (LG, CFG),
(RG, CFG), (RG, LG)}.

For notational purposes the functional composition of basic procedural concepts, p_k's, is denoted by a small letter with no subscript, e.g. $p_k\ p_i$ is denoted by p. If p: d $\longrightarrow$ e and q: e $\longrightarrow$ f, then the functional composition q p:d $\longrightarrow$ f.

Definition III.1.8

$\bar{P}$ = {q | q ε P or q is the composition of elements in P}

Definition III.1.9

$$DOM(p) = \{ \prod_{i=1}^{n} d_i \mid (\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d_j') \varepsilon\ t(p_n) \text{ for } p = p_1\ p_2 \ldots p_n, \text{ and } p \varepsilon \bar{P}\}$$

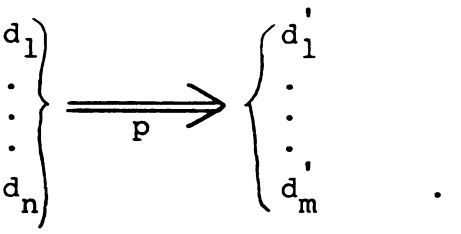
Definition III.1.10

$$RNG(p) = \{ \prod_{j=1}^{m} d_j' \mid (\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d_j') \varepsilon\ t(p_1) \text{ for } p = p_1\ p_2 \ldots p_n, \text{ and } p \varepsilon \bar{P}\}$$

Definition III.1.11 A functional T from $\bar{P}$ to binary relations, called <u>transform</u>, is defined as follows:

$$T(p) = \{(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d_j') \mid d_i, d_j' \varepsilon D \text{ for } 1 \leq i \leq n \text{ and } 1 \leq j \leq m, \text{ and } \prod_{i=1}^{n} d_i \leq dom_k(p) \text{ for } dom_k(p) \varepsilon DOM(p), \text{ and } \prod_{j=1}^{m} d_j' \geq rng_r(p) \text{ for } rng_r(p) \varepsilon RNG(p)\}.$$

Graphically, an element of T(p) is denoted by



.

The relation T is described by t, S, and P. Obviously, $t \subseteq T$.

Example III.1.12 Consider the declarative concepts

D = {context-free grammar(CFG), regular grammar(RG), context-free language(CFL), context-sensitive language(CSL)},

the procedural concept

P = {language determination(LD)},

and the transformation

LD: CFG ⟶ CFL .

(CFG, CFL) is in t(LD) and (RG, CFG) and (CFL, CSL) are in S. By Definition III.1.11

T(LD) = {(CFG, CFL), (RG, CFL), (CFG, CSL), (RG, CSL)}.

The above example illustrates the sub-domain and super-range aspects of Definition III.1.11. The additional mappings identified are; LD: RG ⟶ CFL ; LD: CFG ⟶ CSL ; and LD: RG ⟶ CSL.

Example III.1.13 As an additional example of the functional T, consider the concepts

D = {context-free grammar(CFG), context-sensitive grammar(CSG), derivation sequence(DS), left-most canonical derivation(LMCD), right-most canonical derivation(RMCD), syntax tree(ST), sentential

form(SF), sentence(S)},

P = {sequence structuring(SS), grammar induction(GI)},

and the transformations

SS: DS × CFG ⟶ ST × S

GI: ST × S ⟶ CFG

GI SS: DS × CFG ⟶ CFG .

Then

t(SS) = {(DS × CFG, ST × S)}

t(GI) = {(ST × S, CFG)}

S = {(CFG, CSG), (S, SF), (RMCD DS), (LMCD, DS)}

From t, S, and the functional composition GI SS, the following elements are identified by Definition III.1.11:

T(SS) = {(DS × CFG, ST × S), (RMCD × CFG, ST × S),
(LMCD × CFG, ST × S), (DS × CFG, ST × SF),
(RMCD × CFG, ST × SF), (LMCD × CFG, ST × SF)},

T(GI) = {(ST × S, CFG), (ST × S, CSG)}, and

T(GI SS) = {(DS × CFG, CFG), (LMCD × CFG, CFG),
(RMCD × CFG, CFG), (DS × CFG, CSG),
(LMCD × CFG, CSG), (RMCD × CFG, CSG)}.

Functional composition, sub-domain, and super-range are illustrated in the example. From the initial three mappings an additional eleven mappings have been identified.

Definition III.1.14 The <u>relational composition</u> of T(p) and T(q), denoted as T(p) o T(q), is defined by

$$T(p) \circ T(q) = \{(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d'_j) \mid \exists \prod_{k=1}^{r} d''_k \text{ such that}$$

$$(\prod_{i=1}^{n} d_i, \prod_{k=1}^{r} d''_k) \in T(p) \text{ and } (\prod_{k=1}^{r} d''_k, \prod_{j=1}^{m} d'_j)$$

ε T(q)}.

Relational composition of T is related to functional composition by T(p) o T(q) $\subseteq$ T(q p).

The operation of finding elements of t and S for a given T is at best a guess among many candidates. Let

$t_1(p) = \{(d_2, d_3), (d_2, d_4)\}$, and

$S = \{(d_1, d_2)\}$.

Then

$T_1(p) = \{(d_2, d_3), (d_1, d_3), (d_1, d_4), (d_2, d_4)\}$.

Also let

$t_2(p) = \{(d_2, d_3), (d_1, d_3)\}$, and

$S = \{(d_3, d_4)\}$.

Then

$T_2(p) = \{(d_2, d_3), (d_1, d_3), (d_1, d_4), (d_2, d_4)\}$.

However $T_1(p)$ and $T_2(p)$ are the same.

Definition III.1.15 A pedagogic knowledge representation is a 4-tuple <D, P, t, S>.

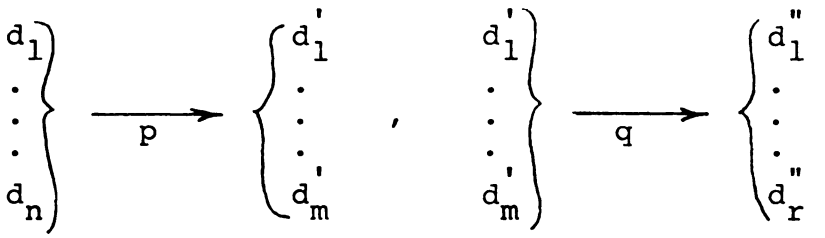
III.2 Diagram Construction

The graphical representation of the relations simple transform and subset is called an elementary diagram. Elementary diagrams are combined to form a diagram. The parts of a diagram that are elementary diagrams are called diagram segments. An operation is a general rule that describes the manner in which the elementary diagrams and diagram segments are manipulated.

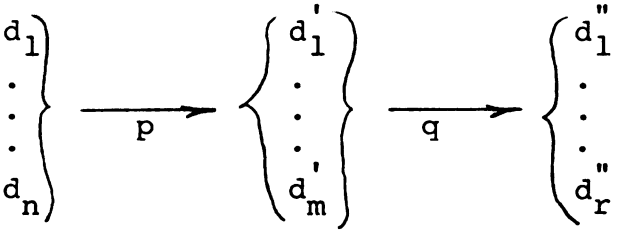
The eight operations below are called construction operations because of the building nature of the operations.

In the description of the eight operations below, the first two diagrams are elementary diagrams or diagram segments; the last diagram is the results of combining the first two diagrams. The eight construction operations are:

Operation III.2.1



yields

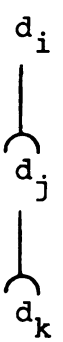


The interpretation of the resulting diagram is that

p: $\prod_{i=1}^{n} d_i \longrightarrow \prod_{j=1}^{m} d'_j$ and that q: $\prod_{j=1}^{m} d'_j \longrightarrow \prod_{k=1}^{r} d''_k$.

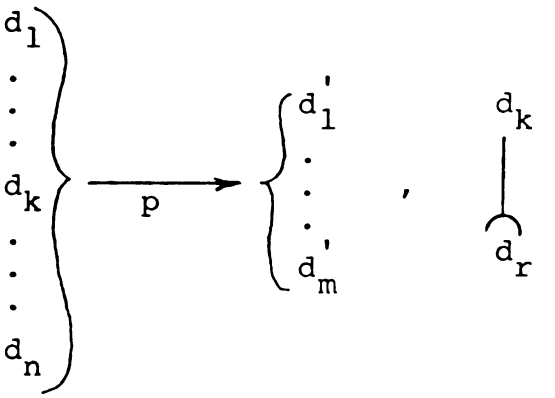
Operation III.2.2

yields

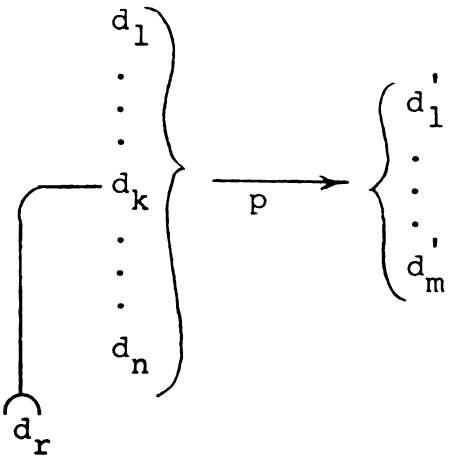


The resulting diagram indicates that $d_k \subset d_j \subset d_i$.

Operation III.2.3



yields



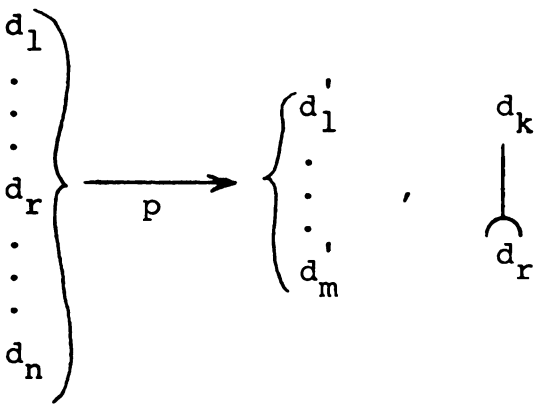
The resulting diagram indicates that

$p: d_1 \times \ldots \times d_k \times \ldots \times d_n \longrightarrow d_1' \times \ldots \times d_m'$

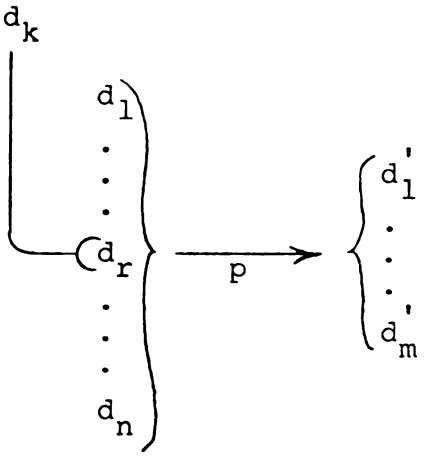
and that

$p: d_1 \times \ldots \times d_r \times \ldots \times d_n \longrightarrow d_1' \times \ldots \times d_m'$.

Operation III.2.4



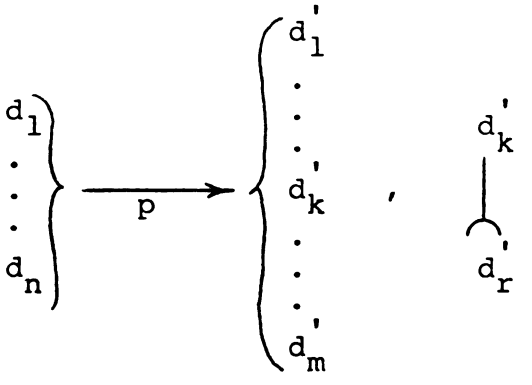
yields



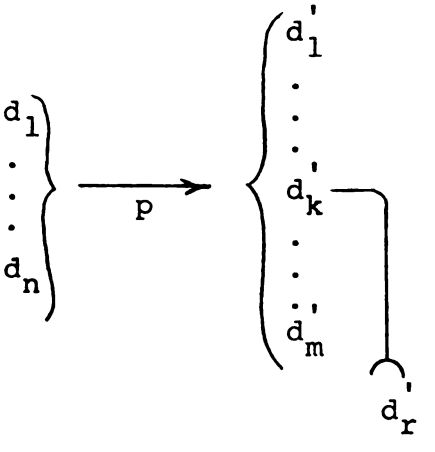
The interpretation of the resulting diagram is that

$p: d_1 \times \ldots \times d_r \times \ldots \times d_n \longrightarrow d_1' \times \ldots \times d_m'$.

Operation III.2.5



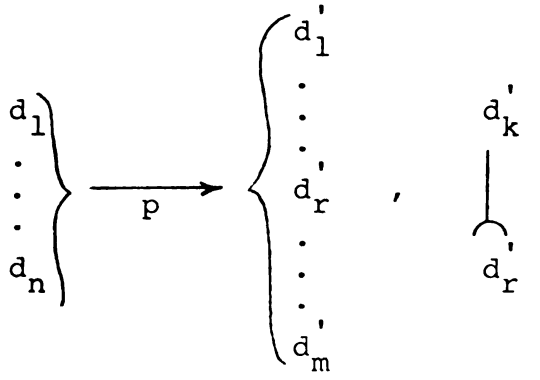
yields



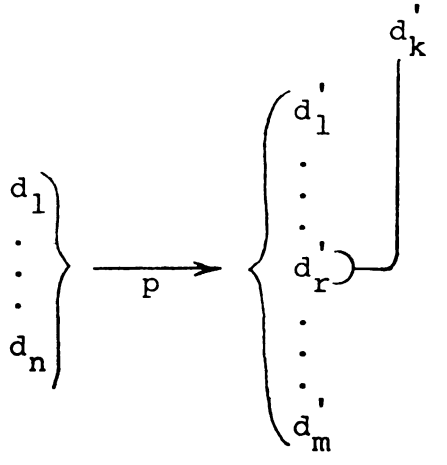
The resulting diagram indicates that

$p: d_1 \times \ldots \times d_n \longrightarrow d_1' \times \ldots \times d_k' \times \ldots \times d_m'$.

Operation III.2.6



yields



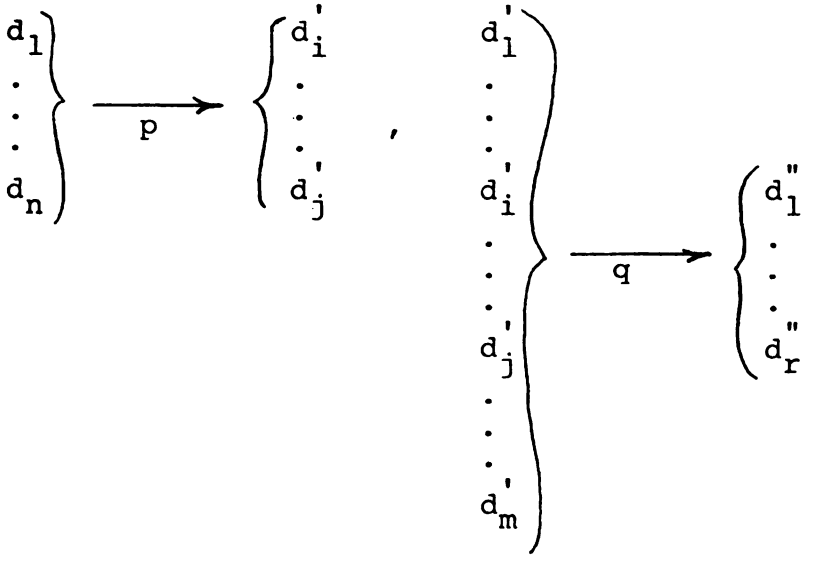
The interpretation of the resulting diagram is that

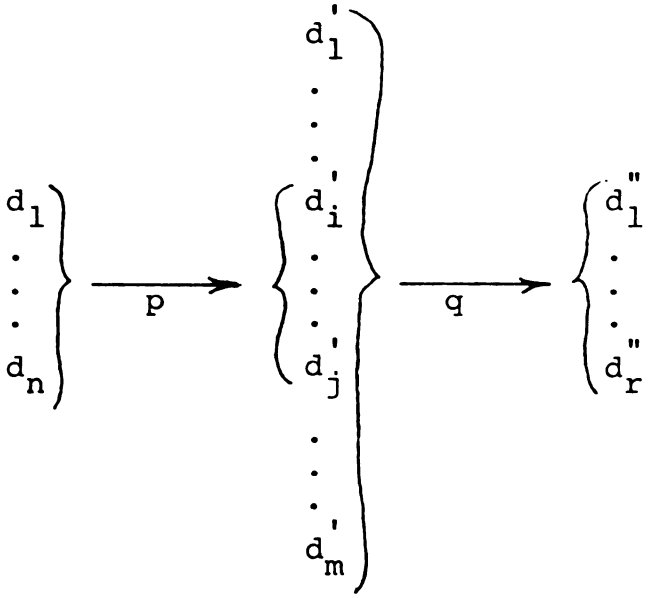
$$p: d_1 \times \ldots \times d_n \longrightarrow d_1' \times \ldots \times d_r' \times \ldots \times d_m'$$

and that

$$p: d_1 \times \ldots \times d_n \longrightarrow d_1' \times \ldots \times d_k' \times \ldots \times d_m' \quad .$$

Operation III.2.7

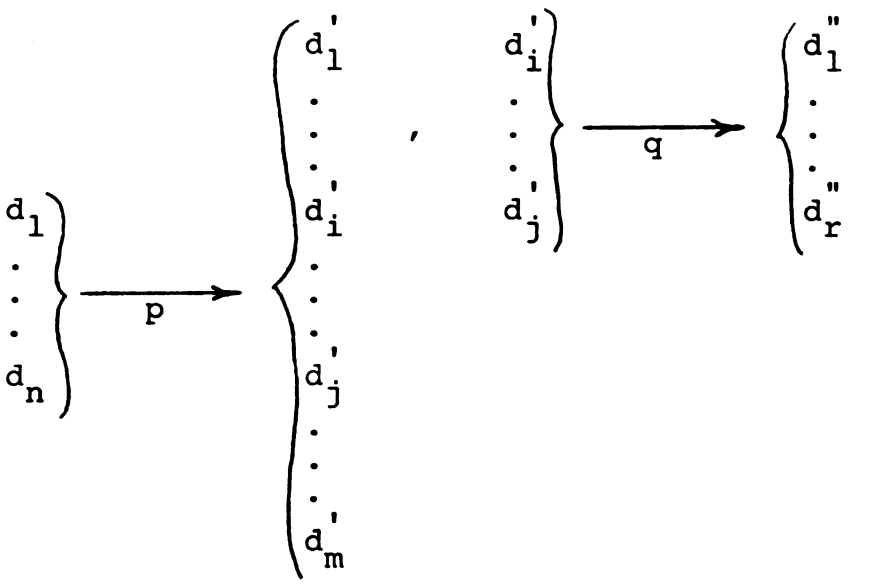


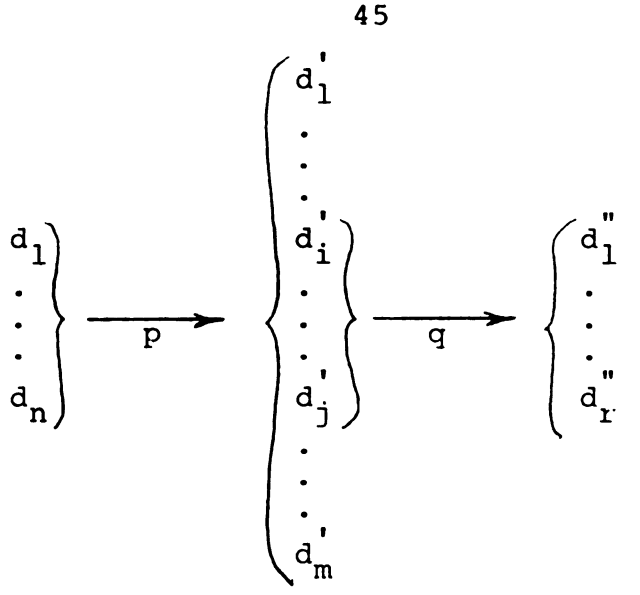
yields 

The resulting diagram indicates that

$$p: \prod_{u=1}^{n} d_u \longrightarrow \prod_{v=i}^{j} d'_v \quad \text{and that} \quad q: \prod_{k=1}^{m} d'_k \longrightarrow \prod_{w=1}^{r} d''_w \ .$$

Operation III.2.8



yields 

The resulting diagram indicates that

$$p: \prod_{u=1}^{n} d_u \longrightarrow \prod_{k=1}^{m} d'_k \quad \text{and that} \quad q: \prod_{v=i}^{j} d'_v \longrightarrow \prod_{w=1}^{r} d''_r \quad .$$

Definition III.2.9 A pedagogic, functional diagram is a diagram constructed by any composition of Operations III.2.1 through III.2.8.

Example III.2.10 Consider the declarative concepts

D = {derivation sequence(DS), sentence(S), sentential form(SF), syntax tree(ST), regular grammar(RG), context-free grammar(CFG)},

the procedural concepts

P = {grammar induction(GI), sequence structuring(SS)},

and the mappings

GI: S × ST ⟶ CFG

and

SS: DS × CFG ⟶ S × ST .

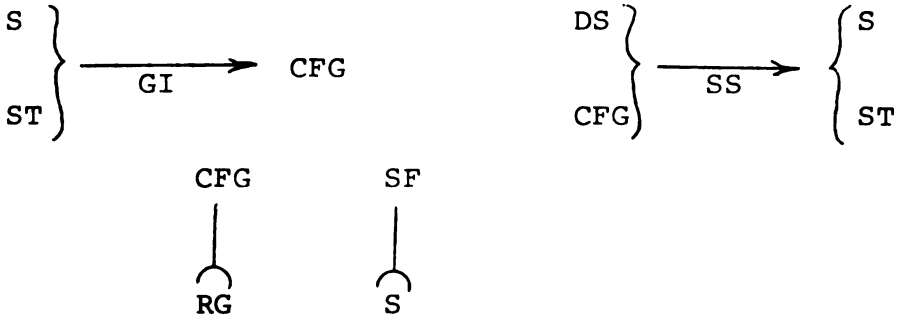
Then

t(GI) = {(S × ST, CFG)}

t(SS) = {(DS × CFG, S × ST)}

S = {(RG, CFG), (S, SF)}

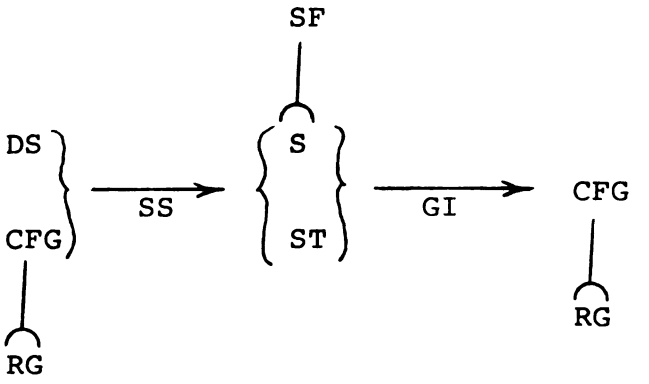
The above relational elements are depicted in Figure III.2.11.



Elementary Pedagogic, Functional Diagrams

Figure III.2.11

The elementary diagrams in Figure III.2.11 are combined by Operations III.2.1 through III.2.8 to result in the diagram in Figure III.2.12.



Pedagogic, Functional Diagram

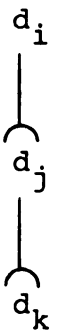
Figure III.2.12

III.3 Diagram Implication

The pedagogic, functional diagrams are used to discover elements in T by the manipulations that are described in this section. The construction of a pedagogic, functional diagram involves combinations of elements from t and S under the Operations III.2.1 through III.2.8. The elements in T that are identified from a given diagram are those elements specified by t and S in the construction of that diagram.

Before presenting operations to identify elements in T, an operation of a reductive nature is presented.

Operation III.3.1 A pedagogic, functional diagram segment of the form



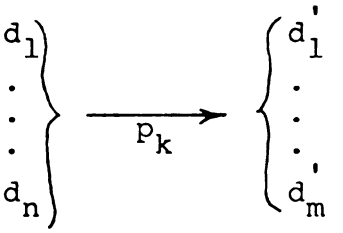
is replaced by the diagram segment

.

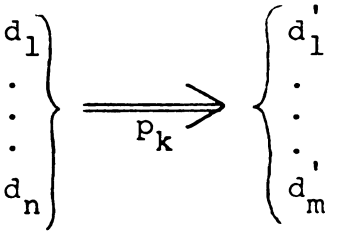
The latter diagram segment represents the relational composition of $S(d_k, d_j)$ and $S(d_j, d_i)$.

The following four operations along with the above operation are used to identify elements in T.

Operation III.3.2 The diagram segment

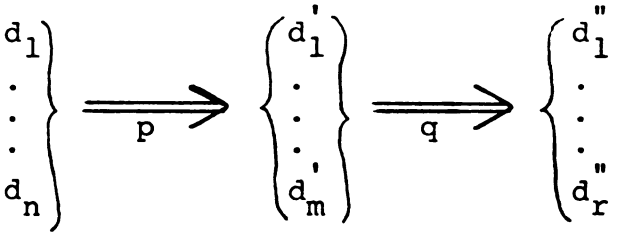


is replaced by

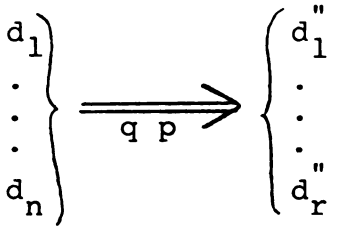


$(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d'_j)$ is an element of $T(p_k)$ since $(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d'_j)$ is an element of $t(p_k)$ and $t(p_k) \subseteq T(p_k)$.

Operation III.3.3 The diagram segment

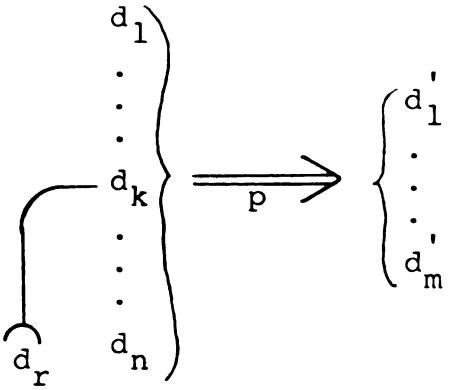


is replaced by

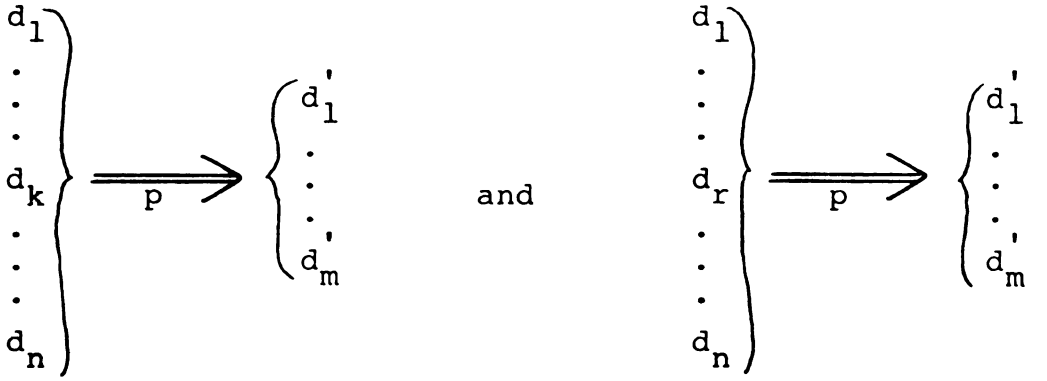


where q and p are elements in $\overline{P}$. $(\prod_{i=1}^{n} d_i, \prod_{k=1}^{r} d''_k)$ is an element of $T(q\ p)$ since $T(p) \circ T(q) \subseteq T(q\ p)$.

Operation III.3.4 The diagram segment

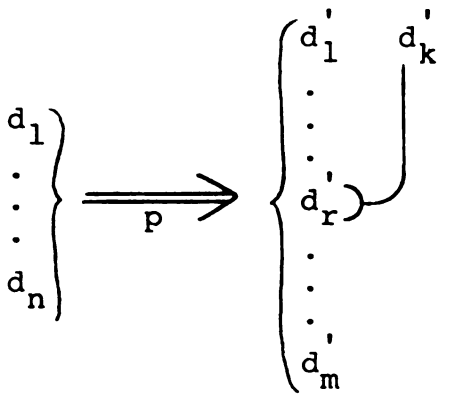


is replaced by the two diagram segments

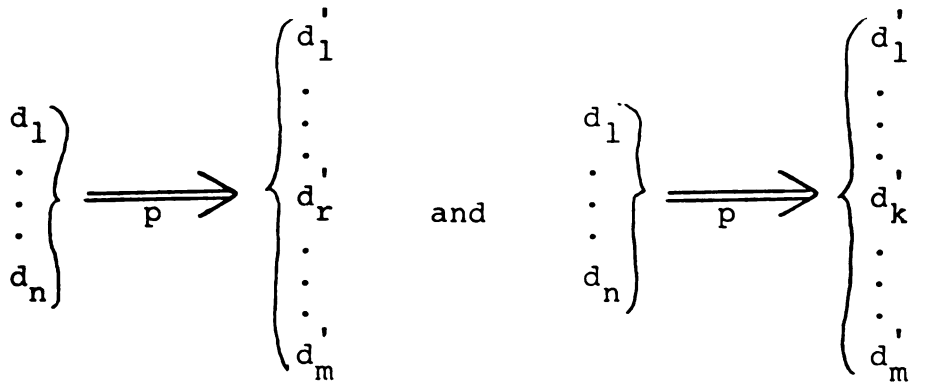


$(d_1 \times \ldots \times d_k \times \ldots \times d_n, d'_1 \times \ldots \times d'_m)$ and $(d_1 \times \ldots \times d_r \times \ldots \times d_n, d'_1 \times \ldots \times d'_m)$ are elements in T(p) by Definition III.1.11. The element $(d_1 \times \ldots \times d_k \times \ldots \times d_n, d'_1 \times \ldots \times d'_m)$ is also discovered in Operation III.3.2.

Operation III.3.5 The diagram segment



is replaced by the two diagram segments

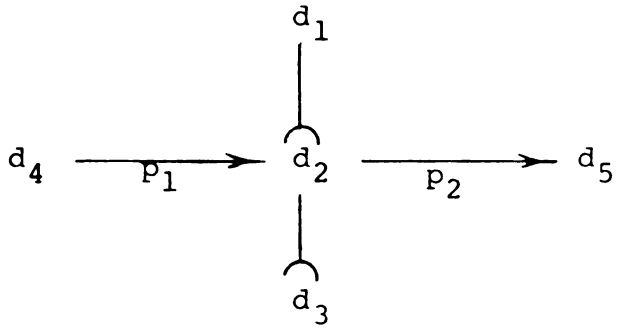


$(d_1 \times \ldots \times d_n, d_1' \times \ldots \times d_r' \times \ldots \times d_m')$ and $(d_1 \times \ldots \times d_n, d_1' \times \ldots \times d_k' \times \ldots \times d_m')$ are elements in T(p) by Definition III.1.11. The element $(d_1 \times \ldots \times d_n, d_1' \times \ldots \times d_r' \times \ldots \times d_m')$ is also discovered in Operation III.3.2.

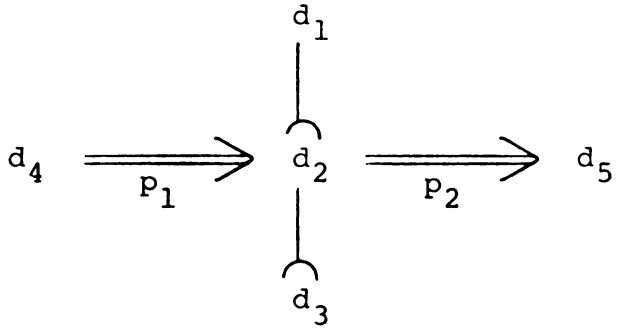
Operations III.3.1 through III.3.5 are called "discovery operations" since each operation either identifies or leads to the identification of new knowledge in C. The new knowledge is either an additional element in S or additional transformations that might not have been in the original representation. The additional transformations are depicted as additional elements in T either as new procedural concepts from Operation III.3.3 or as new domains and/or ranges of procedural concepts from Operations III.3.4 and III.3.5.

Operations III.3.1, III.3.3, III.3.4, and III.3.5 may lead to disjoint diagrams.

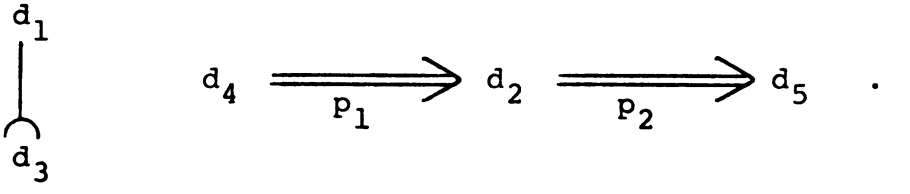
Example III.3.6 Consider the pedagogic, functional diagram



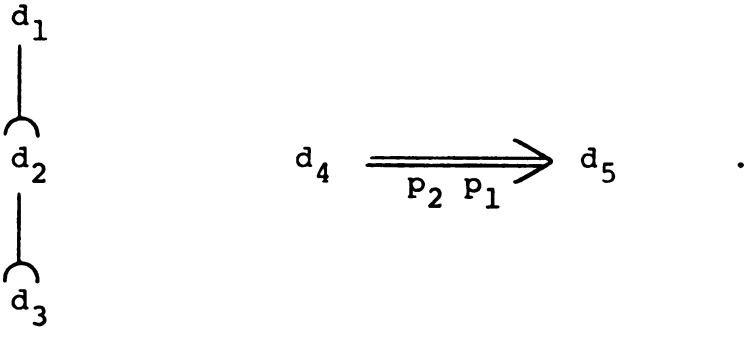
which will result in the diagram



after applying Operation III.3.2 twice. Operation III.3.1 will yield the disjoint diagram



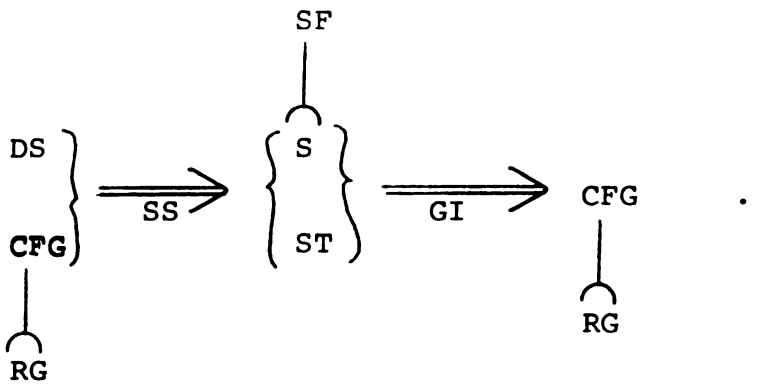
Operation III.3.3 will yield the disjoint diagram



All elements in T that are in a given pedagogic, functional diagram can be discovered by a finite number of compositions of Operations III.3.1 through III.3.5.

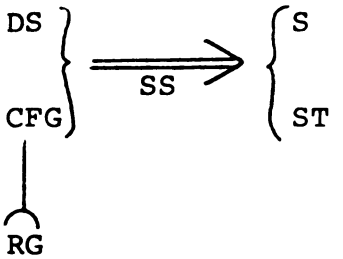
Since all pedagogic, functional diagrams are finite, every combination of operations can be applied to the original diagram yielding every possible element in T that can be discovered by the Operations III.3.1 through III.3.5.

Example III.3.7 Consider the pedagogic, functional diagram in Figure III.2.12. After two applications of Operation III.3.2, the diagram is

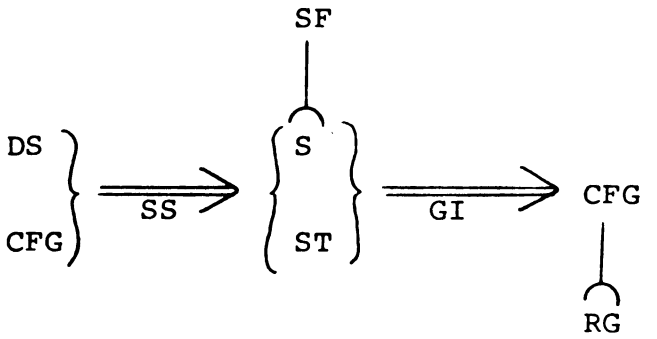


The elements discovered are (DS × CFG, S × ST) ε T(SS) and (S × ST, CFG) ε T(GI).

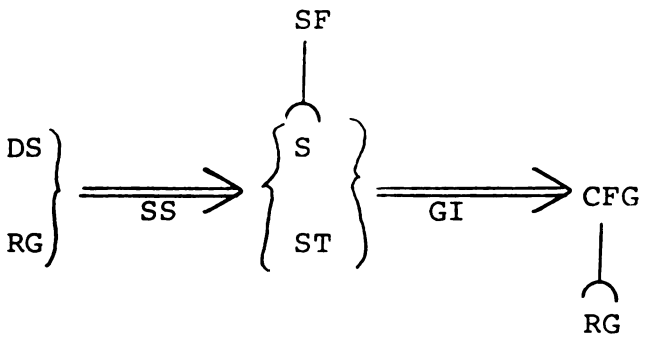
Examining the diagram segment



and applying Operation III.3.4, the diagram segment is replaced by two diagram segments resulting in two diagrams, namely



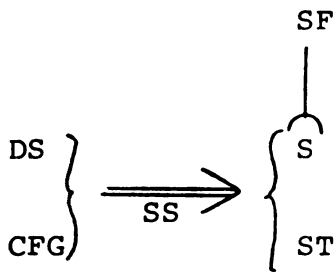
and



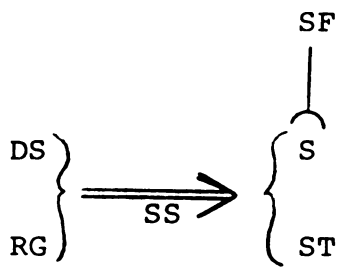
.

The elements (DS × CFG, S × ST) and (DS × RG, S × ST) are discovered to be members of T(SS).

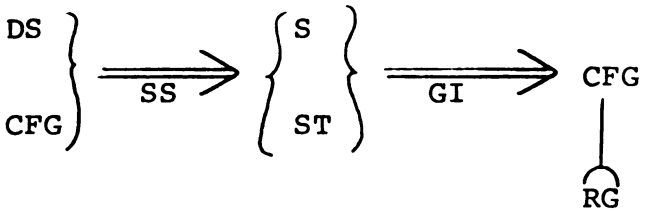
From the diagram segments



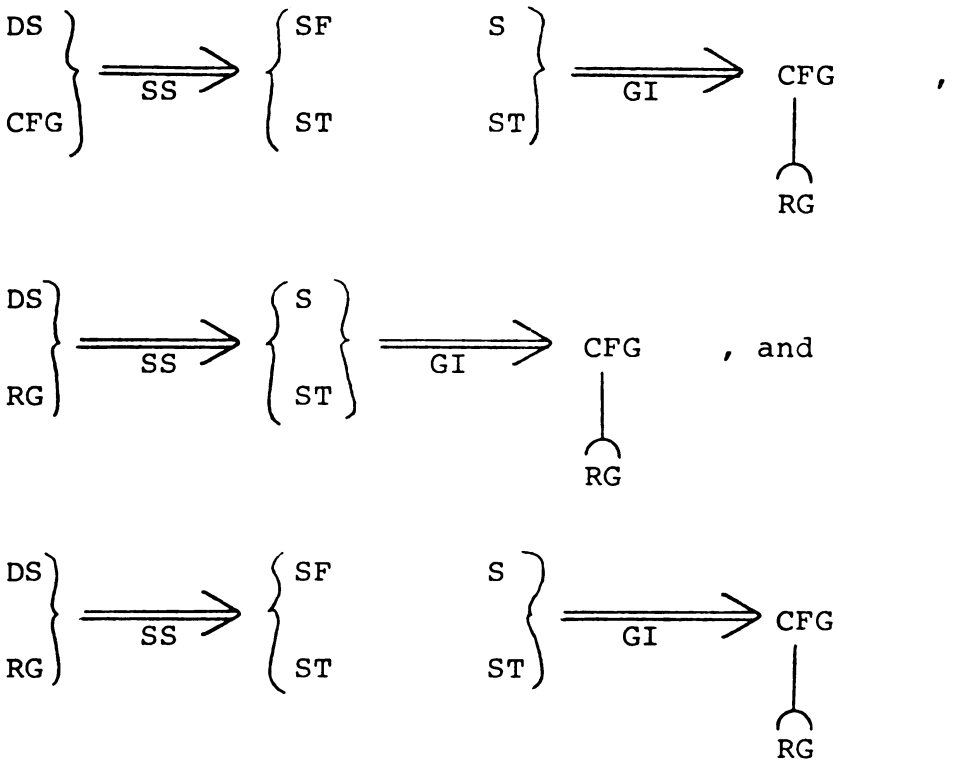
and



under Operation III.3.5, (DS × CFG, S × ST), (DS × CFG, SF × ST), (DS × RG, S × ST), and (DS × RG, SF × ST) are discovered to be elements in T(SS). The diagrams

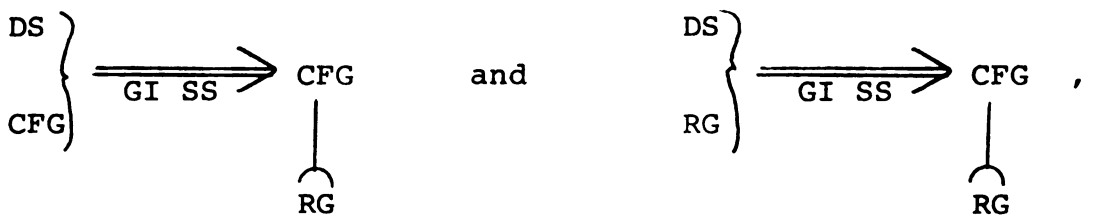


,



are the results of the application of Operation III.3.5. The second and fourth diagrams are disjoint.

From the first and third diagrams, immediately above, the diagrams



respectively, result from Operation III.3.3. (DS × CFG, CFG) and (DS × RG, CFG) are discovered to be elements in T(GI SS).

There are alternative compositions of Operations III.3.1 through III.3.5 that will discover the same elements as the above example. Summarizing the above example, the following elements in T were discovered from Figure III.2.12:

$T(SS) = \{(DS \times CFG, S \times ST), (DS \times RG, S \times ST),$
$(DS \times CFG, SF \times ST), (DS \times RG, SF \times ST)\}$

$T(GI) = \{(S \times ST, CFG)\}$

$T(GI\ SS) = \{(DS \times CFG, CFG), (DS \times RG, CFG)\}$

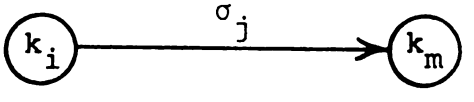
III.4 Functional Automaton

The discovery operations of Section III.3 identify elements in T that are present in a specific pedagogic, functional diagram. All possible diagrams would need to be constructed to identify all elements in T from the given t's and S's. To construct all diagrams would require all possible combinations of Operations III.2.1 through III.2.8 applied to the elements in t and S. Some diagrams might be countably infinite. An alternative description of all elements in T is desirable for computational reasons.

The alternative description that is developed in this section is called a functional automaton.

Definition III.4.1 A finite <u>functional automaton</u> M over an alphabet Σ is a 5-tuple $<K, \Sigma, \delta, q_0, F>$, where K is a finite, non-empty set of states, Σ is a finite input alphabet, δ is a mapping of $K \times \Sigma$ into K, q_0 in K is the initial state, and $F \subseteq K$ is the set of final states.

Notationally, the mapping $\delta: k_i \times \sigma_j \longrightarrow k_m$ is denoted by $\delta(k_i, \sigma_j) = k_m$ and is represented graphically by



where $k_i, k_m \in K$ and $\sigma_j \in \Sigma$. The initial state and final

state are represented graphically by

and

,

respectively.

The domain of δ is extended to $K \times \Sigma^*$, where Σ^* is the set of all strings of finite length composed of symbols from Σ and also including the empty string, by

$$\hat{\delta}(q, \varepsilon) = q$$

$$\hat{\delta}(q, ax) = \delta(\hat{\delta}(q, x), a)$$

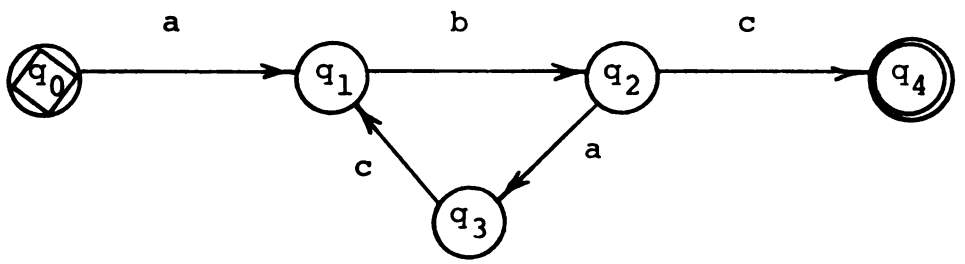
for each $x \in \Sigma^*$, $a \in \Sigma$, $q \in K$, and ε is the empty string. Since δ and $\hat{\delta}$ agree wherever δ is defined, δ will be used for both δ and $\hat{\delta}$ without any confusion.

Definition III.4.2 A <u>sentence</u> x is accepted by M if $\delta(q_0, x) = p$ for some p in F.

Definition III.4.3 The <u>language</u> of M, denoted L(M), is defined by

$L(M) = \{x \mid \delta(q_0, x) \text{ is in } F\}$.

Example III.4.4 Consider the functional automaton M where M is



A sentence in L(M) is "cba." This is shown by

$$\begin{aligned} \delta(q_0, cba) &= \delta(\delta(q_0, ba), c) \\ &= \delta(\delta(\delta(q_0, a), b), c) \\ &= \delta(\delta(q_1, b), c) \end{aligned}$$

$$= \delta(q_2, c)$$

$$= q_4$$

Since $\delta(q_0, cba)$ is in q_4 which is a final state in M, "cba" is a sentence accepted by the automaton. L(M) is $c\ (\ b\ c\ a\ )^*\ b\ a$.

Let dr be a set composed of all domains and ranges of elements in P. Let DR be a set composed of all domains, sub-domains, ranges, and super-ranges of elements in P. DR is computed from dr by Algorithm III.4.5.

Algorithm III.4.5 Domain-Range Extensions

The final w_i is DR.

i. $i = 1$

ii. $w_i = dr$

iii. $i = i + 1$

iv. $w_i = w_{i-1} \cup \{Y \mid Y = y_1 \times \ldots \times d_k \times \ldots \times y_n,$
$Z = y_1 \times \ldots \times d_j \times \ldots \times y_n,\ Z \in w_{i-1},$
$(d_k, d_j) \in S\}$

v. Repeat steps iii and iv until $w_i = w_{i-1}$

vi. $i = i + 1$

vii. $w_i = w_{i-1} \cup \{Z \mid Z = z_1 \times \ldots \times d_j \times \ldots \times z_n,$
$Y = z_1 \times \ldots \times d_k \times \ldots \times z_n,\ Y \in w_{i-1},$
$(d_k, d_j) \in S\}$

viii. Repeat steps vi and vii until $w_i = w_{i-1}$

Step iv computes the sub-domains, and step vii computes the super-ranges. If $(X, Y) \in T(p)$, then X and Y are in DR.

A functional automaton $\langle K, \Sigma, \delta, q_0, F\rangle$ that can be used to identify all functional compositions in C is

defined by

$K = DR,$

$\Sigma = P,$

$q_0 = DR_i$ where $DR_i \in DR,$

$F = \{DR_{j_1}, \ldots, DR_{j_n} \mid DR_{j_k} \in DR$ for

$k = 1, \ldots, n\},$

$\delta(DR_i, p_k) = DR_j$ if $(DR_i, DR_j) \in T(p_k)$ for

$p_k \in P.$

The language of $\langle DR, P, \delta, DR_i, \{DR_{j_1}, \ldots, DR_{j_n}\}\rangle$ is the set of all functional compositions that map DR_i into DR_{j_k} for $k = 1, \ldots, n.$

Example III.4.6 Consider the following relations:

$T(p_1) = \{(D, t)\}$

$T(p_2) = \{(t, D)\}$

$T(p_3) = \{(S \times r, t)\}$

$T(p_4) = \{(s \times r, t)\}$

where the declarative concepts are

D = {angle in degrees(D), angle in radians(t),
radius(r), arc length(s), sector area(S)}.

The procedural concepts p_1, p_2, p_3, and p_4 are the mappings $p_1: D \longrightarrow t$; $p_2: t \longrightarrow D$; $p_3: S \times r \longrightarrow t$; and $p_4 : s \times r \longrightarrow t$, respectively.

dr = {D, t, S × r, s × r}

Since S is empty, DR = dr.

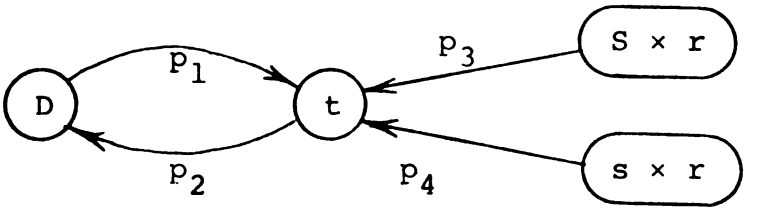
K = {D, t, S × r, s × r}

$\Sigma = \{p_1, p_2, p_3, p_4\}$

$\delta(D, p_1) = t$ $\qquad\qquad$ $\delta(S \times r, p_3) = t$

$\delta(t, p_2) = D$ $\qquad\qquad$ $\delta(s \times r, p_4) = t$

The graphical representation of M is



.

If S × r is the initial state and D is the final state, the language of the automaton is $p_2 (p_1 p_2)^* p_3$. Thus $T(p_2 p_3) = \{(S \times r, D)\}$; $T(p_2 p_1 p_2 p_3) = \{(S \times r, D)\}$; $T(p_2 p_1 p_2 p_1 p_2 p_3) = \{(S \times r, D)\}$; etc. By considering all combinations of elements in DR as initial states and final states of a functional automaton, all elements in T are represented by the functional automaton.

III.5 Non-Repetitive Functional Compositions

For computational reasons and later usage of the functional compositions, a direct calculation of all non-repetitive functional compositions and corresponding elements in T from elements in t and S is developed.

Definition III.5.1 A composition of basic functions $p_{k_1} p_{k_2} \cdots p_{k_n}$ is <u>non-repetitive</u> if and only if for every basic function in the composition $p_{k_i} \neq p_{k_j}$ for $i \neq j$ and $i, j = 1, \ldots, n$.

The remainder of this section develops an algorithmic procedure to calculate all non-repetitive functional compositions.

Let dr be a set composed of all domains and ranges of elements in P. Let DR be a set composed of all domains, sub-domains, ranges, and super-ranges of elements in P. DR is computed by Algorithm III.4.5.

Let $M_{t(p_k)}$ be an n by n function matrix whose rows and columns are ordered identically by the elements in DR and where n is the number of elements in DR. $M_{t(p_k)}$ is defined by

$$(M_{t(p_k)})_{i,j} = \begin{cases} p_k & \text{if } (DR_i, DR_j) \in t(p_k) \\ 0 & \text{otherwise} \end{cases}$$

$p_k + p_r$ as an element of a function matrix means that $p_k: DR_i \longrightarrow DR_j$ and that $p_r: DR_i \longrightarrow DR_j$. The "+" in $p_k + p_r$ is a formal symbol. + is commutative, i.e. $p_k + p_r$ is identical to $p_r + p_k$. + is associative, i.e. $(p_k + p_r) + p_v$ is identical to $p_k + (p_r + p_v)$.

The addition of two function matrices, $M_{t(p_k)}$ and $M_{t(p_r)}$, is denoted as $M_{t(p_k)} + M_{t(p_r)}$ and is defined by

$$(M_{t(p_k)}+M_{t(p_r)})_{i,j} = \begin{cases} p_k + p_r \text{ if } (M_{t(p_k)})_{i,j} = p_k \\ \quad \text{and } (M_{t(p_r)})_{i,j} = p_r \\ p_k \text{ if } (M_{t(p_k)})_{i,j} = p_k \text{ and} \\ \quad (M_{t(p_r)})_{i,j} = 0 \\ p_r \text{ if } (M_{t(p_k)})_{i,j} = 0 \text{ and} \\ \quad (M_{t(p_r)})_{i,j} = p_r \\ 0 \end{cases}$$

Let M_t represent all basic functional transformations. M_t is defined by

$$M_t = \sum_{i=1}^{m} M_{t(p_i)} \quad \text{where P has m elements.}$$

Let M_S be an n by n Boolean matrix whose rows and columns are ordered identically to those of M_t. M_S is defined by

$$(M_S)_{i,j} = \begin{cases} 1 & \text{if } DR_i \subset DR_j \\ 0 & \text{otherwise} \end{cases}$$

Let $\hat{M}_S$ be the transitive closure of M_S(7).

The multiplication of an element from a Boolean matrix, M_S, with an element from a function matrix, M_t, is defined by

$$(M_S)_{i,j}\ (M_t)_{x,y} = \begin{cases} p_k & \text{if } (M_S)_{i,j} = 1 \text{ and } (M_t)_{x,y} = p_k \\ 0 & \text{otherwise} \end{cases}$$

and $(M_S)_{i,j}\ (M_t)_{x,y} = (M_t)_{x,y}\ (M_S)_{i,j}$.

The multiplication of a Boolean matrix, M_S, with a function matrix, M_t, is denoted as $M_S M_t$ or $M_t M_S$ and is defined as usual matrix multiplication.

The basic functions, their domains, sub-domains, ranges, and super-ranges are represented by M_T which is determined by the formula

$$M_T = M_t + \hat{M}_S M_t + M_t \hat{M}_S + \hat{M}_S M_t \hat{M}_S$$

M_t represents all basic functional relations. $\hat{M}_S M_t$ represents all sub-domains of the functional relations. $M_t \hat{M}_s$ represents all super-ranges of the functional relations. $\hat{M}_S M_t \hat{M}_S$ represents all sub-domains, and

super-ranges of the functional relations.

The above formula simplifies via

$$M_T = M_t + M_t\hat{M}_S + \hat{M}_S M_t + \hat{M}_S M_t \hat{M}_S$$

$$= M_t(I + \hat{M}_S) + \hat{M}_S M_t (I + \hat{M}_S) \quad \text{where I is}$$

the identity matrix

$$= M_t \hat{M}_S^R + \hat{M}_S M_t \hat{M}_S^R \quad \text{where } \hat{M}_S^R \text{ is the reflexive}$$

closure of $\hat{M}_S$, i.e. $\hat{M}_S^R = I + \hat{M}_S$

$$= (I + \hat{M}_S) M_t \hat{M}_S^R$$

$$= \hat{M}_S^R M_t \hat{M}_S^R$$

The multiplication of two elements from the function matrices M_1 and M_2 is defined by

$$(M_1)_{i,j}\ (M_2)_{x,y} = \begin{cases} \sum_{i,j} q_j p_i & \text{if } (M_1)_{i,j} = \sum_i p_i \text{ and} \\ & (M_2)_{x,y} = \sum_j q_j \\ 0 & \text{otherwise} \end{cases}$$

The multiplication of two function matrices is defined as usual matrix multiplication.

The non-repetitive functional compositions of basic functions are represented by the expression

$$\sum_{i=1}^{n} M_T^i$$

Each power of M_T represents the number of basic functions in the functional composition. Elements in T are identified immediately from the above expression.

Example III.5.2 Consider the pedagogic, functional

diagram in Figure III.2.11. The figure depicts the elements (DS × CFG, S × ST) ε t(SS), (S × ST, CFG) ε t(GI), and (RG, CFG) and (S, SF) ε S. The domains and ranges are

dr = {DS × CFG, S × ST, CFG}.

From Algorithm III.4.5

DR = {DS × CFG, S × ST, CFG, DS × RG, SF × ST}.

The following matrices are computed:

M_t

	DS × CFG	S × ST	CFG	DS × RG	RG	SF × ST
DS × CFG	0	SS	0	0	0	0
S × ST	0	0	GI	0	0	0
CFG	0	0	0	0	0	0
DS × RG	0	0	0	0	0	0
RG	0	0	0	0	0	0
SF × ST	0	0	0	0	0	0

M_S^R

	DS × CFG	S × ST	CFG	DS × RG	RG	SF × ST
DS × CFG	1	0	0	0	0	0
S × ST	0	1	0	0	0	1
CFG	0	0	1	0	0	0
DS × RG	1	0	0	1	0	0
RG	0	0	1	0	1	0
SF × ST	0	0	0	0	0	1

M_T

	DS × CFG	S × ST	CFG	DS × RG	RG	SF × ST
DS × CFG	0	SS	0	0	0	SS
S × ST	0	0	GI	0	0	0
CFG	0	0	0	0	0	0
DS × RG	0	SS	0	0	0	SS
RG	0	0	0	0	0	0
SF × ST	0	0	0	0	0	0

M_T^2

	DS × CFG	S × ST	CFG	DS × RG	RG	SF × ST
DS × CFG	0	0	GI SS	0	0	0
S × ST	0	0	0	0	0	0
CFG	0	0	0	0	0	0
DS × RG	0	0	GI SS	0	0	0
RG	0	0	0	0	0	0
SF × ST	0	0	0	0	0	0

M_T^3, M_T^4, M_T^5, and M_T^6 are all zero. The non-repetitive compositions are represented by

$$\sum_{i=1}^{6} M_T^i$$

	DS × CFG	S × ST	CFG	DS × RG	RG	SF × ST
DS × CFG	0	SS	GI SS	0	0	SS
S × ST	0	0	GI	0	0	0
CFG	0	0	0	0	0	0
DS × RG	0	SS	GI SS	0	0	SS
RG	0	0	0	0	0	0
SF × ST	0	0	0	0	0	0

The elements in T that can be directly read from the matrix are:

T(SS) = {(DS × CFG, S × ST), (DS × CFG, SF × ST),
(DS × RG, S × ST), (DS × RG, SF × ST)},

T(GI) = {(S × ST, CFG)},

T(GI SS) = {(DS × CFG, CFG), (DS × RG, CFG)}.

The same elements were also discovered under Operations III.3.1 through III.3.5.

III.6 Identification of t and S

The identification of elements of t and S is most easily accomplished from instructional material in the chosen subject area. The instructional material with the greatest wealth of information is homework assignments, examinations, and one or more textbooks in the subject area. The usefulness of the homework assignments and examinations is that the problems represented in the homework assignments and examinations are the basic

functional transformations. These basic functions are the elements in P. The elements in P immediately describe the relation t.

Most declarative concepts will be specified by the basic functions from their domains and/or ranges. Additional declarative concepts will be found in textual material. The textual material will be most helpful in the identification of elements in S.

The procedural and declarative concepts in the subject area "angles" in Section III.7 were discovered from instructional material(13).

Example III.6.1 An example problem(13-p314) is "Find the length of an arc of a circle of radius 6 inches and with a central angle of $\pi/6$ radians." The procedure that solves this problem is found in the back of the text and is to multiply the angle in radians ($\pi/6$) by the radius (6) to obtain the arc length (π). Let p_1 be the above procedure. Then p_1: $r \times t \longrightarrow s$ where r is the radius, t is the angle in radians, and s is the arc length. Obviously, $(r \times t, s) \in t(p_1)$. Usually, the procedures are not given and will have to be identified by the builder of a pedagogic knowledge representation. The declarative concepts "angle in radians," "radius," and "arc length" are identified from the domain and range of p_1.

Examples are another good place to discover procedural and declarative concepts.

Example III.6.2 Example 3(13-p312) in part states

"This means then that an angle of D degrees is $(\pi/180)D$ radians." Applying one's own knowledge about the subject area and some algebra, two procedural concepts are described. The procedures are p_2: D $\longrightarrow$ r and p_3: r $\longrightarrow$ D where r is the angle in radians and D is the angle in degrees. The relational elements (D, t) ε $t(p_2)$ and (r, D) ε $t(p_3)$ are directly identified from p_2 and p_3, respectively. The two declarative concepts "angle in radians" and "angle in degrees" are identified from the domain and range of p_2 and p_3.

Identifying elements in S is a more difficult task. Often an intuitive understanding of the subject area is necessary. From textual material(13-p313) a discussion of the calculation of arc length from the radius and central angle leads to the fact that the circumference is a specific instance of the arc length. Thus (circumference, arc length) ε S.

The preceeding expository in this section describes the identification of elements in D, P, t, and S. Process III.6.3 is not a formal procedure but is an initial approach than an inexperienced builder of a pedagogic knowledge representation might employ.

Process III.6.3 Identifying Elements in t and S

i. Collect all the problems possible in the given subject area. Problems most likely will be found in tests, exams, quizzes, worksheets, homework sets, textbooks, class notes, etc.

ii. Apply any prerequisite knowledge or one's own

knowledge about the subject area to construct additional problems. These additional problems enhance the problems from step i in that they involve new algorithms.

iii. For each problem found in step i and ii, identify a procedure that will solve the stated problem. The identified procedure is a basic function. The procedures form P. The functional relation $t(p_k)$ is immediately specified by the basic function p_k.

iv. Using the declarative concepts in the domains and ranges of the procedures found in step iii and any other declarative concepts in the subject area, identify elements in S.

V. Repeat steps i, ii, iii, and iv until t and S are satisfactory for the purpose at hand.

Process III.6.3 is intended as a guide to building a first pedagogic knowledge representation. As additional pedagogic knowledge representations are constructed, individual techniques will most likely be developed.

III.7 An Example Subject Area

The chosen subject area for this example is "angles" where angles are viewed with respect to circles and some of the other concepts associated with a circle.

The declarative concepts with their abbreviations in parentheses are:

D = {arc length(s), radius(r), angle in radians(t), circumference(c), angle in degrees(D), circle area(A), sector area(S), diameter(d)}.

The procedural concepts in the "angles" subject area are expressed in the form of several formulas. The procedural concepts immediately identified from the given formulas(13) are

$$P = \{p_1: D \longrightarrow t,\ p_2: r \times t \longrightarrow s,\ p_3: c \longrightarrow r,$$
$$p_4: r \longrightarrow A,\ p_5: r \times t \longrightarrow S,\ p_6: d \longrightarrow r\}.$$

The procedural concepts possess no knowledge specific to any subject area. Applying one's own knowledge about the subject area, in this instance algebraic knowledge, additional procedural concepts are identified. As an example of the application of algebraic knowledge in this subject area, consider the algebraic formula to calculate the arc length. The formula is $s = rt$ where r, t, and s are the radius, angle in radians, and arc length, respectively. In algebraic terms if a student was given values for any two of the above three variables, the third variable could be computed. The procedure for computing the arc length, p_2, only computes the arc length. Intuitively, one would also expect the ability to compute the radius from a given angle and arc length. After all, a simple algebraic manipulation of the above formula would make the necessary procedure readily available.

The independence of procedural concepts and subject areas is an extremely beneficial feature of the representation. The beneficial effect is that any assumed knowledge is clearly emphasized and identified in the specification of additional procedural concepts in a subject area.

After applying algebraic manipulations to the above six procedures, an additional eight procedures are identified. The basic procedures for the example subject area are:

$$P = \{p_1: D \rightarrow t,\ p_2: r \times t \rightarrow s,\ p_3: c \rightarrow r,$$
$$p_4: r \rightarrow A,\ p_5: r \times t \rightarrow S,\ p_6: d \rightarrow r,$$
$$p_7: t \rightarrow D,\ p_8: s \times r \rightarrow t,\ p_9: s \times t \rightarrow r,$$
$$p_{10}: r \rightarrow c,\ p_{11}: A \rightarrow r,\ p_{12}: S \times r \rightarrow t,$$
$$p_{13}: S \times t \rightarrow r,\ p_{14}: r \rightarrow d\}.$$

The following relations are immediately identified:

$t(p_1) = \{(D, t)\}$ $\quad$ $t(p_2) = \{(r \times t, s)\}$

$t(p_3) = \{(c, r)\}$ $\quad$ $t(p_4) = \{(r, A)\}$

$t(p_5) = \{(r \times t, S)\}$ $\quad$ $t(p_6) = \{(d, r)\}$

$t(p_7) = \{(t, D)\}$ $\quad$ $t(p_8) = \{(s \times r, t)\}$

$t(p_9) = \{(s \times t, r)\}$ $\quad$ $t(p_{10}) = \{(r, c)\}$

$t(p_{11}) = \{(A, r)\}$ $\quad$ $t(p_{12}) = \{(S \times r, t)\}$

$t(p_{13}) = \{(S \times t, r)\}$ $\quad$ $t(p_{14}) = \{(r, d)\}$

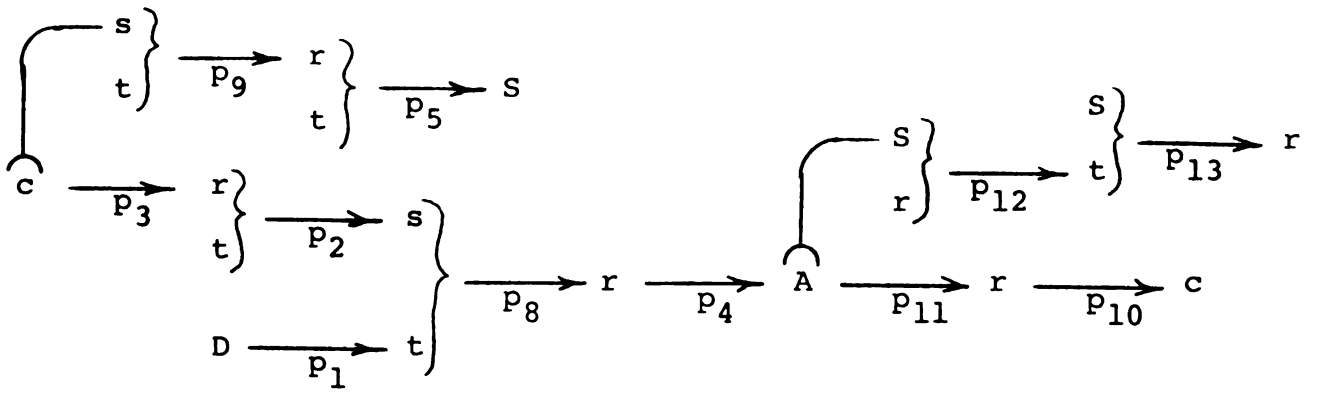
Two elements in S are identified from elements in D.

$$S = \{(A, S), (c, s)\}$$

Many pedagogic, functional diagrams can be constructed under Operations III.2.1 through III.2.8. One of these diagrams in represented in Figure III.7.1. The discovery Operations III.3.1 through III.3.5 can be used to identify elements in T from the diagram in Figure III.7.1. This example will not demonstrate such a process.

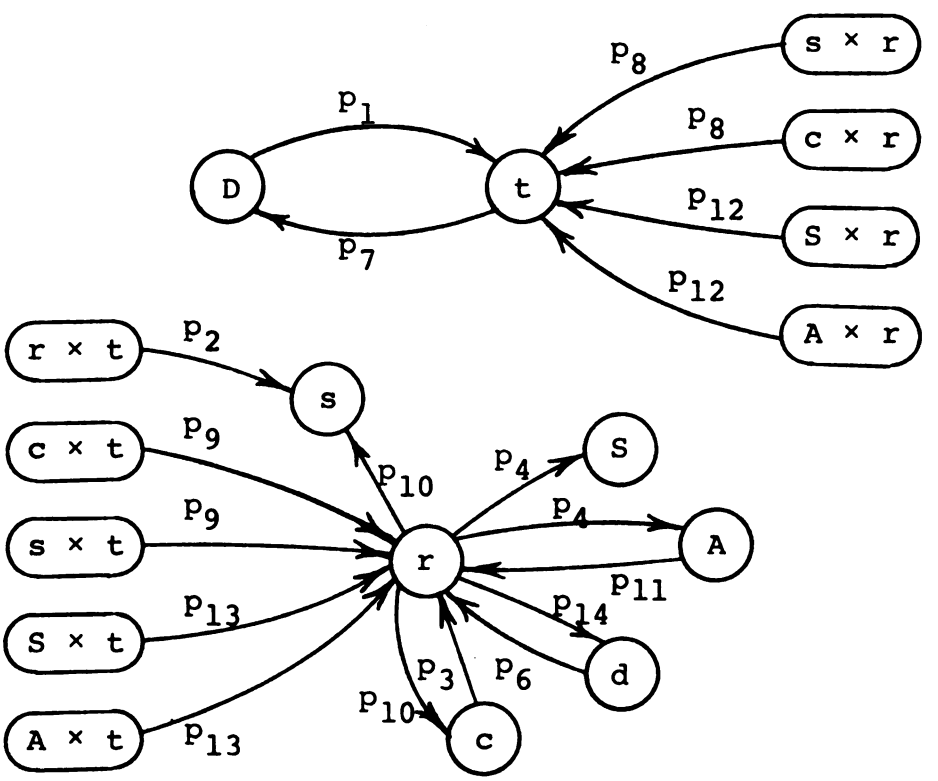
The domains and ranges of P are

$$dr = \{D, t, r \times t, s, c, r, A, S, d, s \times r, s \times t,$$
$$S \times r, S \times t\}.$$



Pedagogic, Functional Diagram of "Angles"

Figure III.7.1



Functional Automaton of "Angles"

Figure III.7.2

Algorithm III.4.5 yields

DR = {D, t, r × t, s, c, r, A, S, d, s × r, s × t,
S × r, S × t, c × r, c × t, A × r, A × t}.

A functional automaton for the subject area "angles" can be constructed. Such an automaton is shown in Figure III.7.2. Functional compositions can be read directly from the automaton. For example, the relation T(p) = {(r, A)} where r is the initial state and A is the final state. The compositions p_4, $p_4p_3p_{10}p_{11}p_4$, $p_4p_6p_{14}$, $p_4p_3p_{10}p_6p_{14}$, and $p_4p_{11}p_4p_{11}p_4$ are some of the possible compositions for p.

All non-repetitive functional compositions for this example are given in Appendix A. The functional automaton in Figure III.7.2 can be used to verify the entries in the matrix. Consider the element that represents all non-repetitive mappings of the diameter of a circle into the circumference of a circle. There are two non-repetitive functional compositions that map "d" into "c," namely $p_{10}p_6$ and $p_{10}p_{11}p_4p_6$. There are fourteen non-repetitive compositions that map the radius of a circle into itself.

CHAPTER IV

PEDAGOGIC REPRESENTATION PROCESSES

The performance of a pedagogic knowledge representation is measured by the representation's ability to determine or display pedagogic aspects, its ability to enumerate problems, and its ability to answer problems. The representation's pedagogic aspects are stated in terms of a pedagogic ordering of concepts. The idea of a connected representation is presented along with an analysis schema to determine the connected parts of a representation.

Several types of problems are identified. A meta-grammar, whose sentential forms represent all possible problems for a given representation, is derived from the elements in the relational T. Complete and extrinsic problems are defined and discussed relative to the meta-grammar.

A solution planning sequence is constructed independent of the problem generation process. Another meta-grammar, whose sentential forms represent all solution planning sequences, is constructed from a representation. The execution of a solution planning sequence is shown to solve the stated problem.

IV.1 Connected Representation

The concepts in a pedagogic representation are connected to other concepts by the elements in t and S. The relation PATH can be thought of as a single step relation in a pedagogic representation.

Definition IV.1.1 A binary relation PATH on C, called pedagogic path, is defined as follows:

$$\text{PATH} = \{(c_i, c_j) \mid 1)\ (c_j, c_i) \in S \text{ for } c_i, c_j \in C, \text{ or}$$

$$2)\ \exists\ d_k, d_r' \text{ such that } \left(\prod_{k=1}^{n} d_k, \prod_{r=1}^{m} d_r'\right) \in t(p_w)$$

for $p_w \in P$, $d_k, d_r' \in D$ for $1 \leq k \leq n$ and $1 \leq r \leq m$, $c_i = d_r'$, and $c_j = d_k$, or

$$3)\ \exists\ d_k, p_w \text{ such that } \left(\prod_{k=1}^{n} d_k, \prod_{r=1}^{m} d_r'\right) \in t(p_w)$$

for $p_w \in P$, $d_k, d_r' \in D$ for $1 \leq k \leq n$ and $1 \leq r \leq m$, $c_j = d_r'$, and $c_i = p_w\}$.

Definition IV.1.2 A connected pedagogic knowledge representation is a set CR which is composed of elements from C such that for every pair of elements in CR, (c_i, c_j), (c_i, c_j) or $(c_j, c_i) \in \widehat{\text{PATH}}$ where $\widehat{\text{PATH}}$ is the transitive closure of PATH.

To find the connected parts of a pedagogic representation, a Boolean matrix M_{PATH} is constructed from PATH. The n rows and columns of M_{PATH} are ordered identically by elements in C where n is the number of elements in C. M_{PATH} is defined by

$$(M_{PATH})_{i,j} = \begin{cases} 1 & \text{if } (c_i, c_j) \in \text{PATH} \\ 0 & \text{otherwise} \end{cases}$$

By taking the transitive closure of the symmetric closure of the reflexive closure(7) of M_{PATH}, denoted by $\widehat{((M_{PATH})^{R})^{S}}$, all undirected possible paths in the pedagogic representation are obtained. The reflexive closure $(M_{PATH})^{R}$ includes each concept on any path emanating from itself. The symmetric closure $((M_{PATH})^{R})^{S}$ makes each path undirected. The transitive closure $\widehat{((M_{PATH})^{R})^{S}}$ includes all concepts that can be reached from a given concept.

If $\widehat{((M_{PATH})^{R})^{S}}$ is all 1's, then there is only one connected pedagogic representation. If the matrix has 0's in it, then more than one connected representation is present. The number of connected representations and the concepts in each are computed by Algorithm IV.1.3.

Algorithm IV.1.3 Connected Representation
The set w_i contains the concepts in the i^{th} connected representation, and i counts the number of connected representations.

i. $i = 0$

ii. $i = i + 1$

iii. Arbitrarily choose a concept from C, call it c_c, such that $(\widehat{((M_{PATH})^{R})^{S}})_{c_c, c_c} = 1$

iv. $w_i = \{c_j \mid (\widehat{((M_{PATH})^{R})^{S}})_{c_c, c_j} = 1 \text{ for all } j\}$

v. $(\widehat{((M_{PATH})^{R})^{S}})_{c_k, c_j} = 0 \text{ for all } c_k, c_j \in w_i$

vi. Repeat steps ii, iii, iv, and v until the matrix is all 0's.

Example IV.1.4 Consider the relations

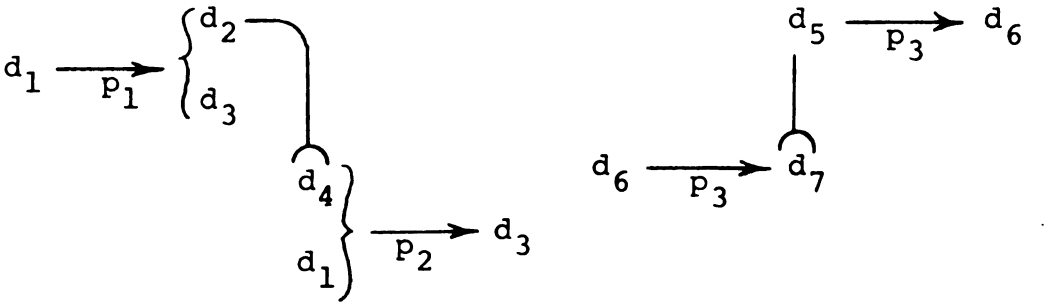
$$t(p_1) = \{(d_1, d_2 \times d_3)\}$$

$$t(p_2) = \{(d_4 \times d_1, d_3)\}$$

$$t(p_3) = \{(d_5, d_6), (d_6, d_7)\}$$

$$S = \{(d_4, d_2), (d_7, d_5)\}$$

A pedagogic, functional diagram composed of the above elements is



The relation PATH is

$$\text{PATH} = \{(d_2, d_4), (d_5, d_7), (p_1, d_2), (p_1, d_3), (d_2, d_1), (d_3, d_1), (p_2, d_3), (d_3, d_4), (p_3, d_6), (p_3, d_7), (d_6, d_5), (d_7, d_6)\}.$$

M_{PATH} is

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	p_1	p_2	p_3
d_1	0	0	0	0	0	0	0	0	0	0
d_2	1	0	0	1	0	0	0	0	0	0
d_3	1	0	0	1	0	0	0	0	0	0
d_4	0	0	0	0	0	0	0	0	0	0
d_5	0	0	0	0	0	0	1	0	0	0
d_6	0	0	0	0	1	0	0	0	0	0
d_7	0	0	0	0	0	1	0	0	0	0

M_{PATH} continued

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	p_1	p_2	p_3
p_1	0	1	1	0	0	0	0	0	0	0
p_2	0	0	1	0	0	0	0	0	0	0
p_3	0	0	0	0	0	1	1	0	0	0

$\widehat{((M_{PATH})^{R})^{S}}$ is

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	p_1	p_2	p_3
d_1	1	1	1	1	0	0	0	1	1	0
d_2	1	1	1	1	0	0	0	1	1	0
d_3	1	1	1	1	0	0	0	1	1	0
d_4	1	1	1	1	0	0	0	1	1	0
d_5	0	0	0	0	1	1	1	0	0	1
d_6	0	0	0	0	1	1	1	0	0	1
d_7	0	0	0	0	1	1	1	0	0	1
p_1	1	1	1	1	0	0	0	1	1	0
p_2	1	1	1	1	0	0	0	1	1	0
p_3	0	0	0	0	1	1	1	0	0	1

Algorithm IV.1.3 yields

$$w_1 = \{d_1, d_2, d_3, d_4, p_1, p_2\}$$

and

$$w_2 = \{d_5, d_6, d_7, p_3\}.$$

The representation contains two connected parts. Thus the above representation is disjoint since it has more than one connected part.

A disjoint representation implies a disjoint subject area. A disjoint subject area suggests that the concepts

in one of the connected representations are not related in any instructional manner to the concepts in other connected representations. It is desirable to have a single connected pedagogic representation where every declarative and procedural concept is related to other declarative and/or procedural concepts in the representation.

IV.2 Pedagogic Dependencies

The elements in t and S can be used to determine if one concept is dependent on another concept.

Definition IV.2.1 A concept c_i is <u>pedagogically dependent</u> on concept c_j if c_j must be mastered before c_i where $c_i, c_j \in C$.

To master a declarative concept means that the general data structure of the declarative concept can be stated and that many specific instances of that general structure can be stated. To master a procedural concept means that the declarative concepts in the domain and range of the procedure have been mastered and that the transformation of many specific domains into their corresponding ranges has been performed.

Definition IV.2.2 A binary relation PD on C, called <u>pedagogic dependency</u>, is defined as follows:

$PD = PATH \cup \{(c_i, c_j) \mid (c_i, c_j) \in S \text{ for } c_i, c_j \in C\}$.

The pedagogic dependency implies that the element $(d_i, d_j) \in t(p_k)$ is mastered in the order d_i, d_j, p_k. The element $(d_i, d_j) \in S$ implies that either d_i, d_j or d_j, d_i are pedagogically dependent on each other. Either d_i or d_j

can be mastered first. The former sequence d_i, d_j is called the bottom-up approach. The latter sequence d_j, d_i is called the top-down approach. The top-down approach introduces the most general declarative concept first followed by more specific declarative concepts. This has the effect of giving the "Big Picture" of a subject area.

A pedagogic dependency matrix M_{PD} is used to find the pedagogically dependent concepts. The n rows and columns of M_{PD} are ordered identically by elements in C. M_{PD} is defined by

$$(M_{PD})_{i,j} = \begin{cases} 1 & \text{if } (c_i, c_j) \in PD \\ 0 & \text{otherwise} \end{cases}$$

If $(M_{PD})_{i,j} = 1$, then c_j is an immediate prerequisite of c_i.

By taking the transitive closure of M_{PD}, $\hat{M}_{PD}$, all pedagogic dependencies in C are obtained. If $(\hat{M}_{PD})_{i,j} = 1$, then c_j is a prerequisite of c_i, Conversely, if $(\hat{M}_{PD})_{i,j} = 1$, then c_i pedagogically depends on c_j. For each concept c_j such that $(\hat{M}_{PD})_{i,j} = 1$, c_j should be presented and mastered before c_i is encountered in the subject area.

$\hat{M}_{PD}$ indicates all prerequisites of a concept, and, conversely, all concepts for which a given concept is a prerequisite. For notational purposes, $(M)_{*,j}$ denotes $(M)_{i,j}$ for all i. $(\hat{M}_{PD})_{*,j}$ indicates all concepts that depend on c_j and all concepts for which c_j is a prerequisite. $(\hat{M}_{PD})_{i,*}$ indicates all concepts upon which c_i is dependent

and all concepts that are a prerequisite of c_i.

M_{PD} is a useful tool in the implementation of a generative instructional system. If a student is allowed to choose the next material that he or she wants to learn in a generative instructional system, M_{PD} and a student's history vector can be used to determine accessibility to the desired concept. The student's history vector is a binary vector where each element represents a concept in the subject area.

Let H be a binary vector of the student's historical accomplishments. H is ordered identically to the ordering of the columns of M_{PD}. H is defined by

$$(H)_i = \begin{cases} 1 & \text{if } c_i \text{ has been mastered by the student} \\ 0 & \text{otherwise} \end{cases}$$

By using an approach factor, either a top-down or a bottom-up approach to presenting the concepts in a subject area can be achieved. Let M_{AF} be a Boolean matrix whose n rows and columns are ordered identically to M_{PD}. For a top-down approach

$$M_{AF} = I \vee M_S^T$$

where I is the identity matrix and M_S^T is the transpose of M_S which is defined in Section III.5. For a bottom-up approach

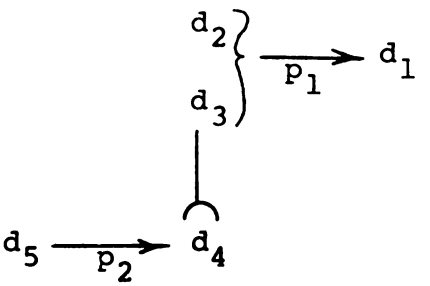
$$M_{AF} = I \vee M_S \quad .$$

If a student wants to next learn and master concept c_i, then he or she would be permitted such access if

$$\bar{H} \wedge \widehat{(M_{PD} \wedge \bar{M}_{AF})}_{i,*}$$

is all zeros where $\bar{H}$ and $\bar{M}_{AF}$ are the complement of H and M_{AF}, respectively. If the conjunction does have concepts not mastered, $\bar{H}$, and that are prerequisite concepts of the choosen concept c_i, $\widehat{(M_{PD} \wedge \bar{M}_{AF})}_{i,*}$, then those precluding concepts are identified as 1's in the conjunction. The precluding concepts could be displayed to the student for a new selection.

Example IV.2.3 Suppose the pedagogic representation is represented by the diagram



Then

$$PD = \{(p_1, d_1), (d_1, d_2), (d_1, d_3), (d_3, d_4), (d_4, d_3), (p_2, d_4), (d_4, d_5)\}$$

M_{PD}

	d_1	d_2	d_3	d_4	d_5	p_1	p_2
d_1	0	1	1	0	0	0	0
d_2	0	0	0	0	0	0	0
d_3	0	0	0	1	0	0	0
d_4	0	0	1	0	1	0	0
d_5	0	0	0	0	0	0	0
p_1	1	0	0	0	0	0	0
p_2	0	0	0	1	0	0	0

From M_{PD} d_2 and d_5 do not have any prerequisite concepts. Assume a bottom-up approach. Then

M_{AF}

	d_1	d_2	d_3	d_4	d_5	p_1	p_2
d_1	1	0	0	0	0	0	0
d_2	0	1	0	0	0	0	0
d_3	0	0	1	0	0	0	0
d_4	0	0	1	1	0	0	0
d_5	0	0	0	0	1	0	0
p_1	0	0	0	0	0	1	0
p_2	0	0	0	0	0	0	1

$\widehat{M_{PD} \wedge \bar{M}_{AF}}$

	d_1	d_2	d_3	d_4	d_5	p_1	p_2
d_1	0	1	1	1	1	0	0
d_2	0	0	0	0	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	1	0	0
d_5	0	0	0	0	0	0	0
p_1	1	1	1	1	1	0	0
p_2	0	0	0	1	1	0	0

The following concepts must be mastered before access to the indicated concept would be permitted:

mastered concepts	access concept
d_5	d_4
d_4, d_5	d_3
d_2, d_3, d_4, d_5	d_1
d_1, d_2, d_3, d_4, d_5	p_1

If H = 0 0 0 0 1 0 0, then a student has mastered only d_5. If the student wants to learn d_4 next, access is permitted to d_4 because the access expression is all 0's. However, if the student wants to learn d_3 next, access is not permitted because the expression contains a 1 and is 0 0 0 1 0 0 0. Thus d_4 needs to be mastered before access is permitted to d_3.

IV.3 Pedagogic Ordering

A pedagogic ordering of concepts in C is a partial ordering. A partial ordering represents an order of presentation of the concepts in the ordering. Such an order of presentation is called an outline. An outline is not intended to represent a best or optimal ordering of concepts.

Definition IV.3.1 A <u>pedagogic partial ordering</u> is the sequence $c_1, c_2, \ldots, c_n$ such that $\exists\, c_j$ for each c_i where $(c_i, c_j) \in PD$ for $c_i, c_j \in C$ and $1 \leq j < i \leq n$.

If for all $c_i, c_j \in C$ and $1 \leq j < i \leq n$ $(c_i, c_j) \in PD$, then the partial ordering $c_1, \ldots, c_n$ is a strict partial ordering. Such a strict partial ordering of a subject area is highly unlikely.

Algorithm IV.3.2 determines a pedagogic partial ordering of concepts that preceed the chosen concept in a pedagogic partial ordering. Algorithm IV.3.5 determines a pedagogic partial ordering of concepts which succeed the chosen concept. Algorithm IV.3.7 combines the two partial orderings to yield an outline containing the

chosen concept.

Algorithm IV.3.2 Pedagogic Dependencies

Let O_1 be a pedagogic partial ordering upon which the arbitrarily chosen concept c_c is pedagogically dependent.

i. Push c_c on the stack and set i = 1.

ii. For each $(p_k, c_c) \in PD$, push p_k on the stack. Push c_c on the stack. Delete the relational element (p_k, c_c) from PD.

iii. For each $(c_c, c_j) \in PD$, push c_j on the stack. Delete the relational element (c_c, c_j) from PD.

iv. i = i + 1

v. If i is greater than the number of elements in the stack, then go to vii, else go to vi.

vi. Set c_c to be the concept at the i^{th} position in the stack. Go to ii.

vii. Pop the top element c_t from the stack. If c_t is not already in O_1, then add c_t to O_1 in the order in which the concepts are popped from the stack.

viii. Repeat step vii until the stack is empty.

Example IV.3.3 The pedagogic dependency relations are

PD = {(12, 2), (2, 1), (1, 2), (2, 3), (3, 2), (4, 2),
(2, 5), (6, 5), (5, 6), (5, 7), (7, 5), (11, 7),
(7, 9), (13, 5), (5, 3), (8, 6), (6, 10)}.

Algorithm IV.3.2 yields the following partial pedagogic orderings for the indicated concepts:

concept	pedagogic partial ordering
8	2, 5, 3, 1, 4, 12, 9, 7, 11, 6, 13, 10, 8
7	2, 1, 3, 5, 4, 12, 10, 6, 8, 7, 13, 9, 11
2	9, 5, 11, 7, 10, 6, 8, 3, 13, 2, 1, 4, 12
2	5, 10, 6, 8, 9, 7, 11, 2, 3, 13, 1, 12, 4

Algorithm IV.3.5 Pedagogically Dependent

Let O_2 be a pedagogic partial ordering of concepts that are pedagogically dependent on the arbitrarily chosen concept c_c.

i. Push c_c on the stack and set i = 1.

ii. For each $(c_j, c_c) \in PD$, push c_j on the stack. Delete the relational element (c_j, c_c) from PD.

iii. i = i + 1

iv. If i is greater than the number of elements in the stack, then go to vi, else go to v.

v. Set c_c to the concept at the i^{th} position in the stack. Go to ii.

vi. Invert the stack making the top of the stack the bottom and the bottom the top.

vii. Pop the top element c_t from the stack. If c_t is not in O_2, add c_t to O_2 in the order in which the concepts are popped from the stack.

viii. Repeat step vii until the stack is empty.

Example IV.3.6 Consider the pedagogic dependency relation in Example IV.3.3. The following pedagogic partial orderings for the chosen concepts via Algorithm IV.3.5 are:

concept	pedagogic partial ordering
8	8
7	7, 5, 2, 6, 13, 12, 1, 3, 4, 8, 11
2	2, 12, 1, 3, 4, 5, 6, 7, 13, 8, 11

The partial orderings that result from Algorithms IV.3.2 and IV.3.5 can be combined to yield a pedagogic partial ordering for all concepts in O_1 and O_2. If O_1 and O_2 contain all the concepts in C, an outline for presenting and mastering the concepts in C can be constructed. Algorithm IV.3.7 computes such an outline.

Algorithm IV.3.7 An Outline

Let O_1 and O_2 be the partial orderings computed by Algorithm IV.3.2 and Algorithm IV.3.5, respectively. Let O be an outline for the concepts in O_1 and O_2.

i. Catenate O_1 and O_2 to form O, $O = O_1, O_2$.

ii. If $O = o_1, \ldots, o_i, c_j, o_{i+2}, \ldots, o_n$, then c_j is deleted from O if $c_j = o_k$ for some $1 \leq k \leq i$.

Example IV.3.8 Consider the pedagogic dependency relation in Example IV.3.3. An outline for the chosen concept 8 or 2 is 2, 5, 3, 1, 4, 12, 9, 7, 11, 6, 13, 10, 8, or 5, 10, 6, 8, 9, 7, 11, 2, 3, 13, 1, 12, 4, respectively.

IV.4 Problems

Definition IV.4.1 A <u>problem</u> is described by the 2-tuple <G, A> where G and A are subsets of D.

G is called the "given" component, and A is called the "asked for" component.

Example IV.4.2 An example of a problem is

<{FORTRAN DO-LOOP program segment}, {FORTRAN IF program segment}> where FORTRAN DO-LOOP program segment is

```
      DO 100 I=1,50
      A(I)=2*I
  100 CONTINUE
```

and FORTRAN IF program segment is

```
      I=0
  200 I=I+1
      A(I)=2*I
      IF(I.LT.50)  GO TO 200        .
```

An English directive for the problem could be "Compute an equivalent FORTRAN IF program segment for the given FORTRAN DO-LOOP program segment given below:

```
      DO 100 I=1,50
      A(I)+2*I
  100 CONTINUE          ."
```

Several types of problems can be identified. Each of these types are useful in determining if a given problem is solvable. A trivial problem is one in which no procedure is required to solve the problem.

Definition IV.4.3 A <u>trivial problem</u> is a problem <G, A> where $\exists\, g_i$ such that for each a_j $(g_i, a_j) \in \hat{S}$ for all $a_j \in A$, $g_i \in G$, and $\hat{S}$ is the transitive closure of S.

In the trivial type of problem the "asked for" is the "given." The "given" is a more specific instance of the more general "asked for." The answer to such a problem is the "given" and requires no computation. Trivial

problems are said to be solvable by the empty procedure. Trivial problems are almost never given in homework assignments or on quizzes.

Definition IV.4.4 A simple problem is a problem $\langle G, A \rangle$ where $\exists \prod_{i=1}^{n} g_i$ such that $(\prod_{i=1}^{n} g_i, \prod_{j=1}^{m} a_j) \in T(p_k)$ where $g_i \in G$ for $1 \leq i \leq n$, $A = \{a_j \text{ for } 1 \leq j \leq m\}$, and $p_k \in P$.

Simple problems can be solved by the application of the one procedure to the declarative concepts in G resulting in the declarative concepts in A. All simple problems can be solved since the solving information is directly encoded in T. As an observation of homework assignments and quizzes, most problems that a student is given are simple problems that require the recognition of the one correct procedure to apply to the "given" data structure.

Definition IV.4.5 A complex problem is a problem $\langle G, A \rangle$ that is not a trivial or simple problem.

Complex problems cannot be solved by the application of a single procedure.

Definition IV.4.6 A problem $\langle G, A \rangle$ is solvable if there exists procedures in P that will map elements in G into all elements in A or if $\langle G, A \rangle$ is a trivial problem.

All simple and trivial problems are solvable. Complex problems may or may not be solvable. If $\langle G, A \rangle$ is solvable, then A is computable from elements in G.

Definition IV.4.7 $\langle G, A \rangle$ is complete if and only if $\langle G, A \rangle$ is solvable.

All simple and trivial problems are complete. Additional analysis is needed to determine if a complex problem is complete. The knowledge for this analysis is present in the pedagogic representation. Consider the problem "Compute the area of a farmer's field knowing that the field is 149 yards long and that the field has 40 cows grazing on it." The pedagogic representation would need to possess the information that the above problem is not complete.

Extra information may be given in a problem to discover if a student can distinguish the extraneous information from the necessary information in solving a problem. Such problems are called extrinsic problems.

Definition IV.4.8 $\langle G, A \rangle$ is extrinsic if and only if $\langle G_1, A \rangle$ is complete where $G_1 \subset G$.

Example IV.4.9 The following problem illustrates an extrinsic problem. "Compute the area of a farmer's field knowing that the field is 147 yards long, that the field has 4 cows grazing under 3 trees, and that the field is 53 yards wide." The answer {area of a field} can be computed from the givens {147 yards long, 53 yards wide} which is a proper subset of {147 yards long, 53 yards wide, 4 cows, 3 trees}.

Extrinsic problems are appropriate problems to give students. The student must learn to discriminate the necessary information from the random noise. This discrimination alters the usual problem solving approach of most students which is to use all information stated

in a problem to solve that problem.

All problems that exist in a pedagogic knowledge representation can be described as the sentential forms of a meta-grammar that is derived from the representation. The meta-grammar describes all possible G's for a given A. Let the meta-grammar $GR_{PROB_A} = \langle V_t, V_n, P_{GR}, S_{GR} \rangle$ where V_t is the set of terminal symbols, V_n is the set of non-terminal symbols, P_{GR} is the set of productions, and S_{GR} is the distinguished symbol. The mapping from a pedagogic knowledge representation to GR_{PROB_A} is

$V_t = \phi$

$V_n = D$

$S_{GR} = a_1 \ldots a_n$ where $A = \{a_i \text{ for } 1 \leq i \leq n\}$

P_{GR} is formed by the mapping

$$\text{For every } \left(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d_j'\right) \in T(p),$$

$$d_j' ::= d_1 \ldots d_n \text{ for } 1 \leq j \leq m.$$

There are no terminal symbols since the givens are described by the sentential forms of GR_{PROB_A}. The m productions that result for every element in T(p) of the form

$\left(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d_j'\right)$ indicate that d_j' is computed from $d_1 \times \ldots \times d_n$ by some procedure.

Let $SF(GR_{PROB_A})$ be the set of sentential forms for GR_{PROB_A} and let sf_i be in $SF(GR_{PROB_A})$. $\langle G, A \rangle$ is complete if for every element g_j in G, g_j is also a symbol in sf_i, and every symbol in sf_i is in G. $\langle G, A \rangle$ is extrinsic if

every symbol in sf_i is in G but there is at least one element g_j in G that is not a symbol in sf_i.

Example IV.4.10 Consider the relations

$$T(p_1) = \{(d_1, d_2)\}$$

$$T(p_2) = \{(d_5 \times d_6 \times d_7, d_4 \times d_1), (d_5 \times d_6 \times d_{10}, d_4 \times d_1)\}$$

$$T(p_3) = \{(d_8, d_6), (d_{11}, d_6)\}$$

$$T(p_4) = \{(d_9, d_{11}), (d_9, d_8)\}$$

The productions of GR_{PROB_A} are:

$d_2 ::= d_1$ $\quad$ $d_4 ::= d_5\ d_6\ d_7$ $\quad$ $d_4 ::= d_5\ d_6\ d_{10}$

$d_6 ::= d_8$ $\quad$ $d_1 ::= d_5\ d_6\ d_7$ $\quad$ $d_1 ::= d_5\ d_6\ d_{10}$

$d_6 ::= d_{11}$ $\quad$ $d_{11} ::= d_9$ $\quad$ $d_8 ::= d_9$

The sentential forms for $GR_{PROB_{\{d_2\}}}$ are

$$SF(GR_{PROB_{\{d_2\}}}) = \{d_2, d_1, d_5\ d_6\ d_7, d_5\ d_6\ d_{10}, d_5\ d_8\ d_7, d_5\ d_{11}\ d_{10}, d_5\ d_9\ d_7, d_5\ d_9\ d_{10}\}.$$

The problem $<\{d_5, d_7, d_{10}\}, \{d_2\}>$ is complete; the problem $<\{d_8, d_{10}\}, \{d_2\}>$ is not complete; and the problem $<\{d_4, d_5, d_9, d_7\}, \{d_2\}>$ is extrinsic.

IV.5 Solutions

The solution process of a pedagogic representation is viewed as a two step process. The first step is the construction of a plan to solve the stated problem. The second step is the execution of that plan. The execution of a plan is viewed as a successive application of procedures to single or multiple declarative concepts which results in declarative concepts that answer the problem.

Definition IV.5.1 A solution planning sequence, denoted by $\mathcal{S}$, for the problem <G, A> is a sequence of declarative and procedural concepts that represent a plan for solving the problem.

$\mathcal{S}$ represents all the procedures and data structures that are necessary to compute the answer for the problem <G, A>. $\mathcal{S}$ is composed of sub-solution planning sequences.

Definition IV.5.2 A sub-solution planning sequence is the sequence $(d_1 \ldots d_n)\ p_k$ where $(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d_j) \in T(p_k)$.

The parentheses are used to indicate all declarative concepts upon which p_k operates. If p_k only operates on a single declarative concept, then the sub-sequence $d_1\ p_k$ is used. The lack of parentheses in the latter case does not result in any ambiguity within a solution planning sequence.

Example IV.5.3 Consider the relations

$$T(p_1) = \{(d_1, d_2)\}$$

$$T(p_2) = \{(d_5 \times d_6 \times d_7, d_1 \times d_4), (d_5 \times d_6 \times d_{10}, d_1 \times d_4)\}$$

$$T(p_3) = \{(d_8, d_6), (d_{11}, d_6)\}$$

$$T(p_4) = \{(d_9, d_{11}), (d_{12}, d_8)\}$$

A solution planning sequence for the problem $<\{d_9\}, \{d_6\}>$ is the sequence $d_9\ p_4\ p_3$. For the problem $<\{d_5, d_9, d_{10}\}, \{d_1, d_4\}>$, $\mathcal{S}$ is $(d_5\ d_9\ p_4\ p_3\ d_{10})\ p_2$.

If <G, A> is complete and not extrinsic, every element in G is also in $\mathcal{S}$, and every declarative concept in $\mathcal{S}$ is also an element in G. Thus G and $\mathcal{S}$ have identical declarative concepts. If <G, A> were extrinsic, then G

would have some declarative concepts that were not in $\mathcal{S}$. If <G, A> is not complete, then $\mathcal{S}$ would not exist by Definition IV.5.1.

Definition IV.5.4 The execution of a sub-solution planning sequence $(d_1 \ldots d_n)\ p_k$, denoted as $\xi((d_1 \ldots d_n)\ p_k)$, is d_j' for some j where $(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d_j') \in T(p_k)$.

If $\mathcal{S} = s_1 \ldots (d_1 \ldots d_n)\ p_k \ldots s_n$, then $\xi(d_1 \ldots d_n)\ p_k$ results in $\mathcal{S} = s_1 \ldots d_j' \ldots s_n$ where d_j' is a declarative concept in the range of p_k. The execution of a sub-solution planning sequence is a non-deterministic process since any declarative concept in the range of p_k can result from the execution of a sub-solution planning sequence.

Definition IV.5.5 The execution of $\mathcal{S}$, $\xi(\mathcal{S})$, is the composition of executions of sub-solution planning sequences.

Example IV.5.6 Using the relations in Example IV.5.3 and the sequence $d_9\ p_4\ p_3$, the sub-solution planning sequence is $d_9\ p_4$. $\xi(d_9\ p_4)$ results in d_{11}. The partially completed solution planning sequence is $d_{11}\ p_3$. $\xi(d_{11}\ p_3)$ yields d_6 which is the answer to the problem $<\{d_9\}, \{d_6\}>$. For the problem $<\{d_5, d_9, d_{10}\}, \{d_1, d_4\}>$, the solution planning sequence is $(d_5\ d_9\ p_4\ p_3\ d_{10})\ p_2\ (d_5\ d_9\ p_4\ p_3\ d_{10})\ p_2$.

The partial progress of a student solving a problem can be monitored. Suppose a student is solving the problem $<\{d_5, d_9, d_{10}\}, \{d_1\}>$. At each step in the

solution of the problem, the student's progress can be monitored by the completion of each sub-problem's solution which corresponds to the execution of a sub-solution planning sequence. Various scenarios could be incorporated in a generative instructional system that uses the execution of solution planning sequences to monitor or guide a student through the solution of a problem.

The identification of $\mathcal{S}$ for a <G, A> is the planning process of solving a problem. $\mathcal{S}$ describes how the computation of the answer will be accomplished. $\mathcal{S}$ is not necessarily unique. There may exist several solution planning sequences for a given problem. The execution of a solution planning sequence is also not necessarily unique. There may be several sub-solution planning sequences in a solution planning sequence that could all be executed next. Thus there may be more than one plan for solving a problem and more than one manner in carrying out that plan.

Definition IV.5.7 A simple solution planning sequence is a solution planning sequence that contains one basic procedural concept.

Definition IV.5.8 A complex solution planning sequence is a solution planning sequence that contains more than one basic procedural concept.

Example IV.5.9 Using the relations in Example IV.5.3, the problem $<\{d_1\}, \{d_2\}>$ has a simple solution planning sequence, namely $d_1\ p_1$. The problem $<\{d_5, d_9, d_{10}\}, \{d_1\}>$ has a complex solution planning sequence, namely $(d_5\ d_9\ p_4\ p_3\ d_{10})\ p_2$.

A solution planning sequence for the problem <G, A> can be derived as a sentential form of a meta-grammar called GR_{SPS_A}. Let $GR_{SPS_A} = \langle V_t, V_n, P_{GR}, S_{GR} \rangle$ where V_t, V_n, P_{GR}, and S_{GR} represent the same grammatical entities as in Section IV.4. V_t, V_n, P_{GR}, and S_{GR} are defined as

$V_t = \bar{P} \cup \{(,)\}$

$V_n = D$

$S_{GR} = a_1 \ldots a_n$ where $A = \{a_1, \ldots, a_n\}$

P_{GR} is constructed via the mapping

$$\text{For every } \left(\prod_{i=1}^{n} d_i, \prod_{j=1}^{m} d_j'\right) \in T(p)$$

$$d_j' ::= (d_1 \ldots d_n)\ p \text{ for } 1 \leq j \leq m.$$

If n = 1, then the parentheses may be omitted from the right hand side of the production.

Example IV.5.10 From Example IV.5.3 the following productions will result from the relations:

$d_2 ::= d_1\ p_1$ $\qquad$ $d_1 ::= (d_5\ d_6\ d_7)\ p_2$

$d_1 ::= (d_5\ d_6\ d_{10})\ p_2$ $\qquad$ $d_4 ::= (d_5\ d_6\ d_7)\ p_2$

$d_4 ::= (d_5\ d_6\ d_{10})\ p_2$ $\qquad$ $d_6 ::= d_8\ p_3$

$d_6 ::= d_{11}\ p_3$ $\qquad$ $d_{11} ::= d_9\ p_4$

$d_8 ::= d_{12}\ p_4$

Each "asked for" has its own meta-grammar that describes all the solution planning sequences for computing that declarative concepts. The sentential forms for GR_{SPS_A}, $SF(GR_{SPS_A})$, are all the solution planning sequences for A. For the problem <G, $\{d_2\}$>

$SF(GR_{SPS_{\{d_2\}}}) = \{d_1\ p_1,\ (d_5\ d_6\ d_7)\ p_2\ p_1,\ (d_5\ d_6\ d_{10})\ p_2\ p_1,$

$(d_5\ d_8\ p_3\ d_7)\ p_2\ p_1,\ (d_5\ d_{11}\ p_3\ d_7)\ p_2\ p_1,$

$(d_5\ d_{12}\ p_4\ p_3\ d_7)\ p_2\ p_1$, $(d_5\ d_9\ p_4\ p_3\ d_7)\ p_2\ p_1\}$

The only simple solution planning sequence for the problem $\langle G, \{d_2\}\rangle$ is $d_1\ p_1$. The other solution planning sequences are complex.

CHAPTER V

CONCLUSION

V.1 Summary

The processes of a pedagogic representation are illustrated by the construction Operations III.2.1 through III.2.8. WLOG the domains and ranges of the procedural concepts are simplified to single elements. Operation III.2.1,

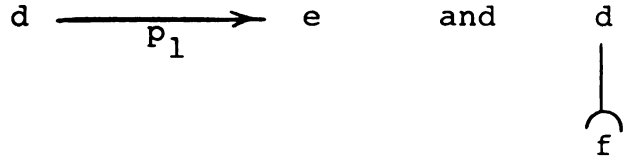
$$d_1 \xrightarrow[p_1]{} d_2 \quad \text{and} \quad d_2 \xrightarrow[p_2]{} d_3$$

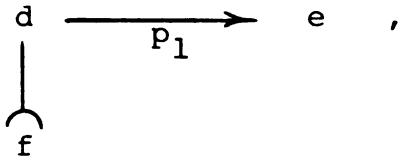
yields $d_1 \xrightarrow[p_1]{} d_2 \xrightarrow[p_2]{} d_3$,

illustrates the combination of two simple problems to yield the complex problem $<\{d_1\}, \{d_3\}>$ whose solution planning sequence is $d_1\ p_1\ p_2$. The implication of the above operation is that the procedural concepts p_1 and p_2 are meaningful in the subject area as the composition $p_2\ p_1$.

Operation III.2.2, III.2.4, and III.2.5 do not yield additional problems but do provide additional pedagogic knowledge about the subject area. These operations help to connect a representation which reflects the connected nature of the subject area.

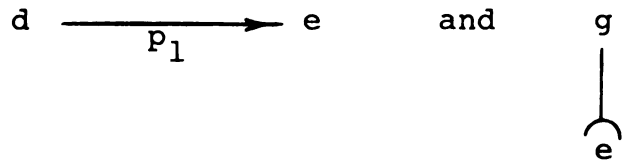
Operation III.2.3,

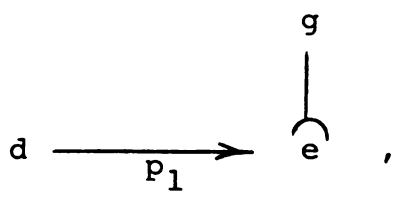


yields 

illustrates the problem <{f}, {e}> in the subject area. Considering the resulting diagram, one may speculate that there exists a subset of e such that p_1 maps f into this subset. Such speculation is strictly subject area dependent and is not treated in this development.

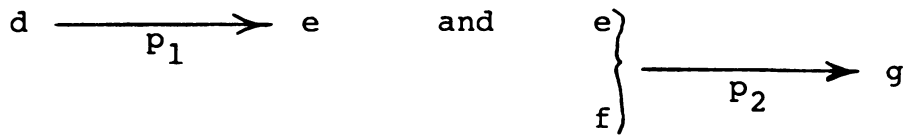
In a similar manner, Operation III.2.6,



yields 

illustrates the problem <{d}, {g}> in the subject area. The last two operations illustrate problems that could be called "mis-named" problems because the domain or range of the problem is mis-named, i.e. d is named f, and e is named g.

Operations III.2.7 and III.2.8,



yields $d \xrightarrow{p_1} \left\{ \begin{matrix} e \\ f \end{matrix} \right\} \xrightarrow{p_2} g$,

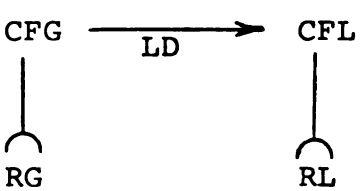
and $d \xrightarrow{p_1} \left\{ \begin{matrix} e \\ f \end{matrix} \right.$ and $f \xrightarrow{p_2} g$

yields $d \xrightarrow{p_1} \left\{ \begin{matrix} e \\ f \xrightarrow{p_2} g \end{matrix} \right.$,

respectively, illustrate problems where the "givens" or "asked fors" are not in the same domain or range of a single procedure. From Operation III.2.7 the problems <{d, f}, {g}> and <{d, f}, {e, g}> are identified. From Operation III.2.8 the problem <{d}, {g}> indicates that the declarative concept e is extraneous to the solution of the problem.

V.2 Extensions

There are several areas in which the results of this work is inadequate. Consider the diagram below:



where

D = {context-free grammar(CFG), context-free language(CFL), regular grammar(RG), regular language(RL)}

and

P = {language determination(LD)}.

The indicated and implied transformations are

LD: CFG $\longrightarrow$ CFL

and

LD: RG $\longrightarrow$ CFL ,

respectively.

Because of the particular subject area, one would like to conclude that

LD: RG $\longrightarrow$ RL .

In general, the above desired conclusion does not hold in all subject areas but does appear to hold in specific instances. The mathematics do not allow the above conclusion. An alternate formulation with perhaps additional information would broaden the possible implications that could be drawn from a given representation. Such an alternate formulation might involve a different treatment of the procedural concepts. The information necessary for the correct implication is imbedded in the procedural concepts. Thus some alternate formulation of the information in the procedural concepts might lead to identifying the appropriate conclusion.

The current work did not consider the components of the basic procedural concepts. An analysis of the basic procedural concepts might lead to a fundamental understanding of their mapping implications, their components, and the appropriate situations for their applicability. Two subject areas with a limited number

of basic procedural concepts could be examined to determine the fundamental components of the procedures. The identification of the fundamental processes might lead to a basic understanding of the two subject areas and a basic understanding of whatever commonality might exist between them.

APPENDIX A

APPENDIX A

Non-repetitive Functional Composition for the Subject Area "Angles"

M_t

		D	t	rxt	s	c	r	A	S	d	sxr	sxt	Sxr	Sxt	cxr	cxt	Axr	Axt
	D	0	p_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	t	p_7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r x	t	0	0	0	p_2	0	0	0	p_5	0	0	0	0	0	0	0	0	0
	s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	c	0	0	0	0	0	p_3	0	0	0	0	0	0	0	0	0	0	0
	r	0	0	0	0	p_{10}	0	p_4	0	p_{14}	0	0	0	0	0	0	0	0
	A	0	0	0	0	0	p_{11}	0	0	0	0	0	0	0	0	0	0	0
	S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	d	0	0	0	0	0	p_6	0	0	0	0	0	0	0	0	0	0	0
s x	r	0	p_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s x	t	0	0	0	0	0	p_9	0	0	0	0	0	0	0	0	0	0	0
S x	r	0	p_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S x	t	0	0	0	0	0	p_{13}	0	0	0	0	0	0	0	0	0	0	0
c x	r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c x	t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A x	r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A x	t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$\hat{M}_S^R$

			D	t	r x t	s	c	r	A	S	d	s x r	s x t	S x r	S x t	c x r	c x t	A x r	A x t
		D	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		t	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	x	t	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		s	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
		c	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
		r	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
		A	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
		S	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
		d	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
s	x	r	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
s	x	t	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
S	x	r	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
S	x	t	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
c	x	r	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
c	x	t	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
A	x	r	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
A	x	t	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

$M_T = \hat{M}_S^R \, M_t \, \hat{M}_S^R$

			D	t	rxt	s	c	r	A	S	d	sxr	sxt	Sxr	Sxt	cxr	cxt	Axr	Axt
		D	0	p_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		t	p_7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	x	t	0	0	0	p_2	0	0	0	p_5	0	0	0	0	0	0	0	0	0
		s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		c	o	o	o	o	o	p_3	0	0	0	0	0	0	0	0	0	0	0
		r	0	0	0	p_{10}	p_{10}	0	p_4	p_4	p_{14}	0	0	0	0	0	0	0	0
		A	0	0	0	0	0	p_{11}	0	0	0	0	0	0	0	0	0	0	0
		S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		d	0	0	0	0	0	p_6	0	0	0	0	0	0	0	0	0	0	0
s	x	r	0	p_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s	x	t	0	0	0	0	0	p_9	0	0	0	0	0	0	0	0	0	0	0
S	x	r	0	p_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	x	t	0	0	0	0	0	p_{13}	0	0	0	0	0	0	0	0	0	0	0
c	x	r	0	p_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	x	t	0	0	0	0	0	p_9	0	0	0	0	0	0	0	0	0	0	0
A	x	r	0	p_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	x	t	0	0	0	0	0	p_{13}	0	0	0	0	0	0	0	0	0	0	0

$\sum_{i=1}^{17} M_{T_n}^{i}$

	D	t	r x t	s	c
D	p_7p_1	p_1	0	0	0
t	p_7	p_1p_7	0	0	0
r x t	0	0	0	p_2	0
s	0	0	0	0	0
c	0	0	0	$p_{10}p_3+p_{10}p_{11}p_4p_3+$ $p_{10}p_6p_{14}p_3+$ $p_{10}p_6p_{14}p_{11}p_4p_3+$ $p_{10}p_{11}p_4p_6p_{14}p_3$	$p_{10}p_3+p_{10}p_{11}p_4p_3+$ $p_{10}p_6p_{14}p_3+$ $p_{10}p_6p_{14}p_{11}p_4p_3+$ $p_{10}p_{11}p_4p_6p_{14}p_3$
r	0	0	0	$p_{10}+p_{10}p_{11}p_4+$ $p_{10}p_6p_{14}+$ $p_{10}p_6p_{14}p_{11}p_4+$ $p_{10}p_{11}p_4p_6p_{14}$	$p_{10}+p_{10}p_{11}p_4+$ $p_{10}p_6p_{14}+$ $p_{10}p_6p_{14}p_{11}p_4+$ $p_{10}p_{11}p_4p_6p_{14}$
A	0	0	0	$p_{10}p_{11}+p_{10}p_6p_{14}p_{11}$	0
S	0	0	0	0	0
d	0	0	0	$p_{10}p_6+p_{10}p_{11}p_4p_6$	0
s x r	p_7p_8	$p_8+p_1p_7p_8$	0	0	0
s x t	0	0	0	$p_{10}p_9+p_{10}p_{11}p_{14}p_9+$ $p_{10}p_6p_{14}p_9+$ $p_{10}p_6p_{14}p_{11}p_4p_9+$ $p_{10}p_{11}p_4p_6p_{14}p_9$	$p_{10}p_9+p_{10}p_{11}p_{14}p_9+$ $p_{10}p_6p_{14}p_9+$ $p_{10}p_6p_{14}p_{11}p_4p_9+$ $p_{10}p_{11}p_4p_6p_{14}p_9$
S x r	p_7p_{12}	$p_{12}+p_1p_7p_{12}$	0	0	0
S x t	0	0	0	$p_{10}p_{13}+p_{10}p_{11}p_4p_{13}$ $+p_{10}p_6p_{14}p_{13}+$ $p_{10}p_6p_{14}p_{11}p_4p_{13}+$ $p_{10}p_{11}p_4p_6p_{14}p_{13}$	$p_{10}p_{13}+p_{10}p_{11}p_4p_{13}$ $+p_{10}p_6p_{14}p_{13}+$ $p_{10}p_6p_{14}p_{11}p_4p_{13}+$ $p_{10}p_{11}p_4p_6p_{14}p_{13}$

$\sum_{i=1}^{17} M_{T_n}^{i}$ continued

	D	t	r x t	s	c
c x r	p_7p_8	$p_8+p_1p_7p_8$	0	0	0
c x t	0	0	0	$p_{10}p_9+p_{10}p_{11}p_4p_9$ $+p_{10}p_6p_{14}p_9+$ $p_{10}p_6p_{14}p_{11}p_4p_9+$ $p_{10}p_{11}p_4p_6p_{14}p_9$	$p_{10}p_9+p_{10}p_{11}p_4p_9$ $+p_{10}p_6p_{14}p_9+$ $p_{10}p_6p_{14}p_{11}p_4p_9+$ $p_{10}p_{11}p_4p_6p_{14}p_9$
A x r	p_7p_{12}	$p_{12}+p_1p_7p_{12}$	0	0	0
A x t	0	0	0	$p_{10}p_{13}+p_{10}p_{11}p_4p_{13}$ $+p_{10}p_6p_{14}p_{13}+$ $p_{10}p_6p_{14}p_{11}p_4p_{13}+$ $p_{10}p_{11}p_4p_6p_{14}p_{13}$	$p_{10}p_{13}+p_{10}p_{11}p_4p_{13}$ $+p_{10}p_6p_{14}p_{13}+$ $p_{10}p_6p_{14}p_{11}p_4p_{13}+$ $p_{10}p_{11}p_4p_6p_{14}p_{13}$

$\sum_{i=1}^{17} M_{T_n}^{i}$ continued

	r	
	D	0
	t	0
r x	t	0
	s	0
	c	$p_3+p_{11}p_4p_3+p_6p_{14}p_3+p_6p_{14}p_{11}p_4p_3+p_{11}p_4p_6p_{14}p_3$
	r	$p_3p_{10}+p_{11}p_4+p_6p_{14}+p_{11}p_4p_3p_{10}+p_6p_{14}p_3p_{10}+p_3p_{10}p_{11}p_4+$ $p_6p_{14}p_{11}p_4+p_3p_{10}p_6p_{14}+p_{11}p_4p_6p_{14}+p_6p_{14}p_{11}p_4p_3p_{10}+$ $p_{11}p_4p_6p_{14}p_3p_{10}+p_6p_{14}p_3p_{10}p_{11}p_4+p_3p_{10}p_6p_{14}p_{11}p_4+$ $p_{11}p_4p_3p_{10}p_6p_{14}+p_3p_{10}p_{11}p_4p_6p_{14}$
	A	$p_{11}+p_3p_{10}p_{11}+p_6p_{14}p_{11}+p_6p_{14}p_3p_{10}p_{11}+p_3p_{10}p_6p_{14}p_{11}$
	S	0
	d	$p_6+p_3p_{10}p_6+p_{11}p_4p_6+p_{11}p_4p_3p_{10}p_6+p_3p_{10}p_{11}p_4p_6$
s x	r	0
s x	t	$p_9+p_3p_{10}p_9+p_{11}p_4p_9+p_{11}p_4p_3p_{10}p_9+p_6p_{14}p_3p_{10}p_9+$ $p_6p_{14}p_9+p_3p_{10}p_{11}p_4p_9+p_6p_{14}p_{11}p_4p_9+p_3p_{10}p_6p_{14}p_9+$ $p_{11}p_4p_6p_{14}p_9+p_6p_{14}p_{11}p_4p_3p_{10}p_9+p_6p_{14}p_3p_{10}p_{11}p_4p_9+$ $p_{11}p_4p_3p_{10}p_6p_{14}p_9+p_3p_{10}p_{11}p_4p_6p_{14}p_9$
S x	r	0
S x	t	$p_{13}+p_{11}p_4p_{13}+p_6p_{14}p_{13}+p_3p_{10}p_{13}+p_6p_{14}p_{11}p_4p_{13}+$ $p_{11}p_4p_6p_{14}p_{13}+p_{11}p_4p_3p_{10}p_{13}+p_6p_{14}p_3p_{10}p_{13}+$ $p_3p_{10}p_6p_{14}p_{13}+p_3p_{10}p_{11}p_4p_{13}+p_6p_{14}p_{11}p_4p_3p_{10}p_{13}+$ $p_{11}p_4p_6p_{14}p_3p_{10}+p_6p_{14}p_3p_{10}p_{11}p_4p_{13}+p_3p_{10}p_6p_{14}p_{11}p_4p_{13}+$ $p_{11}p_4p_3p_{10}p_6p_{14}p_{13}+p_3p_{10}p_{11}p_4p_6p_{14}p_{13}$
c x	r	0

$\sum_{i=1}^{17} M_{T_n}^{i}$ continued

r

$$p_9+p_3p_{10}p_9+p_{11}p_4p_9+p_{11}p_4p_3p_{10}p_9+p_6p_{14}p_3p_{10}p_9+$$
$$p_6p_{14}p_9+p_3p_{10}p_{11}p_4p_9+p_6p_{14}p_{11}p_4p_9+p_3p_{10}p_6p_{14}p_9+$$
$$p_{11}p_4p_6p_{14}p_9+p_6p_{14}p_{11}p_4p_3p_{10}p_9+p_6p_{14}p_3p_{10}p_{11}p_4p_9+$$
$$p_{11}p_4p_3p_{10}p_6p_{14}p_9+p_3p_{10}p_{11}p_4p_6p_{14}p_9$$

c x t

A x r 0

$$p_{13}+p_{11}p_4p_{13}+p_6p_{14}p_{13}+p_3p_{10}p_{13}+p_6p_{14}p_{11}p_4p_{13}+$$
$$p_{11}p_4p_6p_{14}p_{13}+p_{11}p_4p_3p_{10}p_{13}+p_6p_{14}p_3p_{10}p_{13}+$$
$$p_3p_{10}p_6p_{14}p_{13}+p_3p_{10}p_{11}p_4p_{13}+p_6p_{14}p_{11}p_4p_3p_{10}p_{13}+$$
$$p_{11}p_4p_6p_{14}p_3p_{10}+p_6p_{14}p_3p_{10}p_{11}p_4p_{13}+p_3p_{10}p_6p_{14}p_{11}p_4p_{13}+$$
$$p_{11}p_4p_3p_{10}p_6p_{14}p_{13}+p_3p_{10}p_{11}p_4p_6p_{14}p_{13}$$

A x t

$\sum_{i=1}^{17} M_{T_n}^i$ continued

		A	S	d
	D	0	0	0
	t	0	0	0
r x	t	0	p_5	0
	s	0	0	0
	c	$p_4p_3+p_4p_6p_{14}p_3$	$p_4p_3+p_4p_6p_{14}p_3$	$p_{14}p_3+p_{14}p_{11}p_4p_3$
	r	$p_4+p_4p_3p_{10}+p_4p_6p_{14}$ $+p_4p_6p_{14}p_3p_{10}+$ $p_4p_3p_{10}p_6p_{14}$	$p_4+p_4p_3p_{10}+p_4p_6p_{14}$ $+p_4p_6p_{14}p_3p_{10}+$ $p_4p_3p_{10}p_6p_{14}$	$p_{14}+p_{14}p_3p_{10}+p_{14}p_{11}p_4$ $+p_{14}p_{11}p_4p_3p_{10}+$ $p_{14}p_3p_{10}p_{11}p_4$
	A	$p_4p_{11}+p_4p_3p_{10}p_{11}+$ $p_4p_6p_{14}p_{11}+$ $p_4p_6p_{14}p_3p_{10}p_{11}+$ $p_4p_3p_{10}p_6p_{14}p_{11}$	$p_4p_{11}+p_4p_3p_{10}p_{11}+$ $p_4p_6p_{14}p_{11}+$ $p_4p_6p_{14}p_3p_{10}p_{11}+$ $p_4p_3p_{10}p_6p_{14}p_{11}$	$p_{14}p_{11}+p_{14}p_3p_{10}p_{11}$
	S	0	0	0
	d	$p_4p_6+p_4p_3p_{10}p_6$	$p_4p_6+p_4p_3p_{10}p_6$	$p_{14}p_6+p_{14}p_3p_{10}p_6+$ $p_{14}p_{11}p_4p_6+$ $p_{14}p_{11}p_4p_3p_{10}p_6+$ $p_{14}p_3p_{10}p_{11}p_4p_6$
s x	r	0	0	0
s x	t	$p_4p_9+p_4p_3p_{10}p_9+$ $p_4p_6p_{14}p_9+$ $p_4p_6p_{14}p_3p_{10}p_9+$ $p_4p_3p_{10}p_6p_{14}p_9$	$p_4p_9+p_4p_3p_{10}p_9+$ $p_4p_6p_{14}p_9+$ $p_4p_6p_{14}p_3p_{10}p_9+$ $p_4p_3p_{10}p_6p_{14}p_9$	$p_{14}p_9+p_{14}p_3p_{10}p_9+$ $p_{14}p_{11}p_4p_9+$ $p_{14}p_{11}p_4p_3p_{10}p_9+$ $p_{14}p_3p_{10}p_{11}p_4p_9$
S x	r	0	0	0
S x	t	$p_4p_{13}+p_4p_6p_{14}p_{13}+$ $p_4p_3p_{10}p_{13}+$ $p_4p_6p_{14}p_3p_{10}p_{13}+$ $p_4p_3p_{10}p_6p_{14}p_{13}$	$p_4p_{13}+p_4p_6p_{14}p_{13}+$ $p_4p_3p_{10}p_{13}+$ $p_4p_6p_{14}p_3p_{10}p_{13}+$ $p_4p_3p_{10}p_6p_{14}p_{13}$	$p_{14}p_{13}+p_{14}p_{11}p_4p_{13}+$ $p_{14}p_3p_{10}p_{13}+$ $p_{14}p_{11}p_4p_3p_{10}p_{13}+$ $p_{14}p_3p_{10}p_{11}p_4p_{13}$

$\sum_{i=1}^{17} M_{T_n}^{i}$ continued

	A	S	d
c x r	0	0	0
c x t	$p_4p_9+p_4p_3p_{10}p_9+$ $p_4p_6p_{14}p_9+$ $p_4p_6p_{14}p_3p_{10}p_9+$ $p_4p_3p_{10}p_6p_{14}p_9$	$p_4p_9+p_4p_3p_{10}p_9+$ $p_4p_6p_{14}p_9+$ $p_4p_6p_{14}p_3p_{10}p_9+$ $p_4p_3p_{10}p_6p_{14}p_9$	$p_{14}p_9+p_{14}p_3p_{10}p_9+$ $p_{14}p_{11}p_4p_9+$ $p_{14}p_{11}p_4p_3p_{10}p_9+$ $p_{14}p_3p_{10}p_{11}p_4p_9$
A x r	0	0	0
A x t	$p_4p_{13}+p_4p_6p_{14}p_{13}+$ $p_4p_3p_{10}p_{13}+$ $p_4p_6p_{14}p_3p_{10}p_{13}+$ $p_4p_3p_{10}p_6p_{14}p_{13}$	$p_4p_{13}+p_4p_6p_{14}p_{13}+$ $p_4p_3p_{10}p_{13}+$ $p_4p_6p_{14}p_3p_{10}p_{13}+$ $p_4p_3p_{10}p_6p_{14}p_{13}$	$p_{14}p_{13}+p_{14}p_{11}p_4p_{13}+$ $p_{14}p_3p_{10}p_{13}+$ $p_{14}p_{11}p_4p_3p_{10}p_{13}+$ $p_{14}p_3p_{10}p_{11}p_4p_{13}$

$\sum_{i=1}^{17} M_{T_n}^{i}$ continued

		s x r	s x t	S x r	S x t	c x r	c x t	A x r	A x t
	D	0	0	0	0	0	0	0	0
	t	0	0	0	0	0	0	0	0
r x	t	0	0	0	0	0	0	0	0
	s	0	0	0	0	0	0	0	0
	c	0	0	0	0	0	0	0	0
	r	0	0	0	0	0	0	0	0
	A	0	0	0	0	0	0	0	0
	S	0	0	0	0	0	0	0	0
	d	0	0	0	0	0	0	0	0
s x	r	0	0	0	0	0	0	0	0
s x	t	0	0	0	0	0	0	0	0
S x	r	0	0	0	0	0	0	0	0
S x	t	0	0	0	0	0	0	0	0
c x	r	0	0	0	0	0	0	0	0
c x	t	0	0	0	0	0	0	0	0
A x	r	0	0	0	0	0	0	0	0
A x	t	0	0	0	0	0	0	0	0

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Brown, John Seely, and Burton, Richard R., "SOPIE - A Pragmatic Use of Artificial Intelligence in CAI," Proceedings ACM 1974 Annual Conference, 1974, pp 571-579.

2. Carbonell, Jaime R., "AI in CAI: An Artificial Intelligence Approach to Computer Assisted Instruction," IEEE Transactions on Man-Machine Systems, vol MMS-11, no 4, December, 1970, pp 181-189.

3. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," Communications of ACM, vol 13, no 6, June, 1970, pp 377-387.

4. Cohen, Ellis S., "Problems, Mechanisms and Solutions," Technical Report, Department of Computer Science, Carnegie-Mellon University, August, 1976.

5. deKleer, Johan, "Multiple Representations of Knowledge in a Mechanics Problem-Solver," Proceedings of IJCAI-77, August, 1977, pp 299-304.

6. Erman, Lee D., and Lesser, Victor R., "A Multi-level Organization for Problem Solving Using Many, Diverse Cooperating Sources of Knowledge," Technical Report, Department of Computer Science, Carnegie-Mellon University, March, 1975.

7. Gries, David, Compiler Construction for Digital Computers, John Wiley & Sons, 1971.

8. Izquierdo, F. J., "A Generator/Grader of Problems about Syntax of Programming Languages to be used in an Automated Exam System," Department of Computer Science Report UIUCDCS-R-75-755, University of Illinois at Urbana-Champaign, September, 1975.

9. Koffman, Elliot B., "A Generative CAI Tutor for Computer Science Concepts," Proceedings AFIPS Spring Joint Computer Conference, 1972, pp 379-389.

10. _____, and Perry, James, "An Intelligent CAI Monitor and Generative Tutor," Final Report on Project No. 020193 Grant No. OEG-0-72-0895, June, 1975.

11. Minsky, Marvin, "A Framework for Representing Knowledge," MIT Artificial Intelligence Laboratory Memo No. 306, June, 1974.

12. Moore, J., and Newell, A., "How can Merlin Understand?," Technical Report, Department of Computer Science, Carnegie-Mellon University, 1973.

13. Munem, Mustafa ., and Yizzle, James P., Functional Approach to Precalculus, Worth Publishers, 1970.

14. Newell A., and Simon, H. A., Human Problem Solving, Prentice Hall, 1972.

15. Nilsson, N. J., Problem Solving Methods in Artificial Intelligence, McGraw-Hill, 1971.

16. _____, "Artificial Intelligence," Stanford Research Institute Technical Note 89, 1974.

17. Notes from an Interdisciplinary Workshop on Theoretical Issues in Natural Language Processing, June, 1975.

18. Novak, Gordon S., "Computer Understanding of Physics Problems Stated in Natural Languages," Technical Report NL-30, Computer Science Department, The University of Texas at Austin, 1976.

19. _____, "Representations of Knowledge in a Program for Solving Physics Problems," Proceedings of IJCAI-77, August, 1977, pp 286-291.

20. Page, C. V., Greenberg, L. H., and Gates, G. W., "Notes for Introduction to Compiling," 1971.

21. Proceedings of the Theoretical Issues in Natural Language Processing, Association for Computational Linguistics, June, 1975.

22. Prosser, Franklin, and Jensen, Donald D., "Computer Generated Repeatable Tests," Proceedings AFIPS Spring Joint Computer Conference, 1971, pp 295-301.

23. Quillian, M. R., "The Teachable Language Comprehender: A Simulation Program and Theory of Language," Communications of ACM, vol 12, no 8, August, 1969, pp 459-476.

24. Ramani S., and Newell, A., "On the Generation of Problems," Technical Report, Department of Computer Science, Carnegie-Mellon University, November, 1973.

25. Raphael, B., "SIR: A Computer Program for Semantic Information Retrieval," Project MAC Report TR-2, MIT, 1964.

26. Rychener, Michael D., "The Student Production System: A Study of Encoding Knowledge in Production Systems," Technical Report, Department of Computer Science, Carnegie-Mellon University, October, 1975.

27. _____, "Production Systems as a Programming Language for Artificial Intelligence Applications Vol I," Technical Report, Department of Computer Science, Carnegie-Mellon University, December, 1976.

28. Sacerdoti, Earl D., "Procedural Net for Representing Hierarchial Plans," Artificial Intelligence - Research and Applications Progress Report, Contract DAHC04-75-C-0005, May, 1975.

29. Schank, Roger C., Goldman, Niel M., Reiger, Charles J., and Riesbeck, Christopher K., "Inference and Paraphrase by Computer," Journal of ACM, vol 22, no 3, July, 1975, pp 309-328.

30. Simon, R., "Some Relations between Predicate Calculus and Semantic Net Representation of Discourse," Technical Report NL-2, Department of Computer Science, University of Texas at Austin, June, 1971.

31. _____, and Slocum, J., "Generating English Discourse from Semantic Networks," Communications of ACM, vol 15, no 10, October, 1972, pp 891-905.

32. Sussman, Gerald Jay, "Electrical Design A Problem for Artificial Intelligence Research," Proceedings of IJCAI-77, August, 1977, pp 894-900.

33. Uhr, L., "Teaching Machine Programs that Generate Problems as a Function of Interaction with Students," Proceedings of the 24th ACM National Conference, 1969, pp 125-134.

34. Vere, Steven A., "Induction of Relational Productions in the Presence of Background Information," Proceedings of IJCAI-77, August, 1977, pp 349-355.

35. Vickers, F. D., "Cognitive and Creative Test Generators," Proceedings AFIPS Fall Joint Computer Conference, 1972, pp 649-659.

36. Whitlock, L. R., "Interactive Test Construction and Administration in the Generative Exam System," Department of Computer Science Report UIUCDCS-R-76-821, University of Illinois at Urbana-Champaign, August, 1976.

37. Winograd, Terry, "Five Lectures on Artificial Intelligence," Artificial Intelligence Laboratory Memo AIM No. 246, Computer Science Department, Stanford University, September, 1974.

38. Woods, W. A., "Transition Network Grammars for Natural Language Analysis," Communications of ACM, vol 13, no 10, October, 1970, pp 591-606.

39. _____, Kaplan, R. A., and Nash-Webber, B., "The Lunar Sciences Natural Language Information System, BBN Report No 2378, June, 1972.

40. _____, "What's in a Link: Foundations for Semantic Networks," Representation and Understanding: Studies in Cognitive Science, edited by Bobrow and Collins, Academic Press, 1975.