ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

EXECUTIVE EDITORS

Jack Belzer *Albert G. Holzman* *Allen Kent*

VOLUME 9

R
73.87072
B454
9:1

ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

EXECUTIVE EDITORS

Jack Belzer *Albert G. Holzman* *Allen Kent*

UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA

VOLUME 9

Generative Epistemology to Laplace Transforms

MARCEL DEKKER, INC. NEW YORK and BASEL

[illegible] 75/07

MARCEL DEKKER, INC.
270 Madison Avenue, New York, New York 10016

LIBRARY OF CONGRESS CATALOG CARD NUMBER: 74-29436
ISBN: 0-8247-2259-2

Current printing (last digit):
10 9 8 7 6 5 4 3 2 1

PRINTED IN THE UNITED STATES OF AMERICA

CONTRIBUTORS TO VOLUME 9

MASANAO AOKI, Professor, Department of System Science, School of Engineering and Applied Science, University of California, Los Angeles, California: *Gradient Methods*

JACK BELZER, University of Pittsburgh, Pittsburgh, Pennsylvania: *Information Theory*

DAVID CASASENT, Department of Electrical Engineering, Carnegie-Mellon University, Pittsburgh, Pennsylvania: *Holography and Optical Data Processing*

H. JOHN CAULFIELD, Aerodyne Research Inc., Bedford, Massachusetts: *Holography and Optical Data Processing*

MARY R. DEMELIM, Executive Director, Institute of Management Sciences Providence, Rhode Island: *Institute of Management Sciences*

ROGER W. EHRICH, Associate Professor, Department of Computer Science, Virginia Polytechnic Institute and State University., Blacksburg, Virginia: *Handwriting Recognition*

NICHOLAS V. FINDLER, Professor of Computer Science and Mathematics, Computer Science Department, State University of New York at Buffalo, Amherst, New York: *Heuristic Methods and Programming*

WALTER C. GIFFIN, Department of Industrial and Systems Engineering, The Ohio State University, Columbus, Ohio: *Laplace and Geometric Transforms*

GEOFFREY GORDON, IBM Corporation, New York, New York: *GPSS*

WILLIAM H. GRUBER, President, Research and Planning Institute, Inc., West Newton, Massachusetts, and Lecturer, Alfred P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts: *Global and National Computer-Based Models*

LEWIS F. HANES, Manager, Human Sciences, Westinghouse Research and Development, Pittsburgh, Pennsylvania: *Human Factors Engineering*

DAN A. HINSLEY, Human Sciences, Westinghouse Research and Development, Pittsburgh, Pennsylvania: *Human Factors Engineering*

C. A. HOLLINGSWORTH, Chemistry Department, University of Pittsburgh, Pittsburgh, Pennsylvania: *Hamiltonian*

CONTRIBUTORS TO VOLUME 9

V. HUNGERBUHLER, University of Geneva, Geneva, Switzerland: *High Energy Physics*

SANG M. LEE, Chairman and Professor, Department of Management, College of Business Administration, The University of Nebraska, Lincoln, Nebraska: *Goal Programming*

G. JACK LIPOVSKI, Department of Electrical Engineering, College of Engineering, The University of Texas, Austin, Texas: *Hardware Description Languages*

IRWIN C. MARIN, Interdisciplinary Department of Information Science, University of Pittsburgh, Pittsburgh, Pennsylvania: *Generative Epistemology of Problem Solving*

RONALD W. MAY, Department of Geology, The University of Alberta, Edmonton, Alberta, Canada: *Geology*

MARLIN H. MICKLE, Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania: *Hybrid Computers*

STEVEN NAHMIAS, Department of Industrial Engineering, Systems Management Engineering and Operations Research, University of Pittsburgh, Pittsburgh, Pennsylvania: *Inventory Models*

SUZANNE C. NUSS, Interdisciplinary Department of Information Science, University of Pittsburgh, Pittsburgh, Pennsylvania: *Generative Epistemology of Problem Solving*

DON T. PHILLIPS, Department of Industrial Engineering, Texas A & M University, College Station, Texas: *Geometric Programming*

SEYMOUR V. POLLACK, Department of Computer Science, Washington University, St. Louis, Missouri: *High Level Language Programming*

ROBERT W. RECTOR, Executive Director, American Federation of Information Processing Societies, Inc., Montvale, New Jersey: *IFIP (International Federation for Information Processing)*

LARRY J. SHUMAN, Department of Industrial Engineering and Health Operations Research, University of Pittsburgh, Pittsburgh, Pennsylvania: *Health Systems*

THEODOR D. STERLING, Computing Science Program, Simon Fraser University, Burnaby, British Columbia, Canada: *High Level Language Programming* and *Humanizing Information Systems*

CONTRIBUTORS TO VOLUME 9

DAVID STODOLSKY, Center for Research in Management Science, University of California, Berkeley, California: *Group Conferencing with Automatic Mediation*

PETER G. SUTTERLIN, Department of Geology, The University of Western Ontario, London, Ontario, Canada: *Geology*

HAMDY A. TAHA, Professor, Department of Industrial Engineering, University of Arkansas, Fayetteville, Arkansas: *Integer Programming*

JULIA THOMPSON, Department of Physics, University of Pittsburgh, Pittsburgh, Pennsylvania: *High Energy Physics*

HARVEY WOLFE, Department of Industrial Engineering and Health Operations Research, University of Pittsburgh, Pittsburgh, Pennsylvania: *Health Systems*

CONTENTS OF VOLUME 9

Contributors to Volume 9 v

GENERATIVE EPISTEMOLOGY OF PROBLEM SOLVING *Irwin C. Marin in collaboration with Suzanne C. Nuss* 1

GEOLOGY *Peter G. Sutterlin and Ronald W. May* 27

GEOMETRIC PROGRAMMING *Don T. Phillips* 56

GLOBAL AND NATIONAL COMPUTER-BASED MODELS *William H. Gruber* 68

GOAL PROGRAMMING *Sang M. Lee* 83

GPSS *Geoffrey Gordon* 105

GRADIENT METHODS *Masanao Aoki* 118

GROUP CONFERENCING WITH AUTOMATIC MEDIATION *David Stodolsky* 143

HAMILTONIAN *C. A. Hollingsworth* 171

HANDWRITING RECOGNITION *Roger W. Ehrich* 180

HARDWARE DESCRIPTION LANGUAGES *G. Jack Lipovski* 198

HEALTH SYSTEMS *Larry J. Shuman and Harvey Wolfe* 218

HEURISTIC METHODS AND PROGRAMMING *Nicholas V. Findler* 232

HIGH ENERGY PHYSICS *Julia Thompson and V. Hungerbuhler* 236

HIGH LEVEL LANGUAGE PROGRAMMING *Seymour V. Pollack and Theodor D. Sterling* 257

HOLOGRAPHY AND OPTICAL DATA PROCESSING *H. John Caulfield and David Casasent* 284

HUMAN FACTORS ENGINEERING *Lewis F. Hanes and Dan A. Hinsley* 351

HUMANIZING INFORMATION SYSTEMS *Theodor D. Sterling* 361

HYBRID COMPUTERS *Marlin H. Mickle* 371

IFIP (INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING) *Robert W. Rector* 382

INFORMATION THEORY *Jack Belzer* 390

INSTITUTE OF MANAGEMENT SCIENCES *Mary R. DeMelim* 415

CONTENTS OF VOLUME 9

INTEGER PROGRAMMING *Hamdy A. Taha* 419
INVENTORY MODELS *Steven Nahmias* 447
LAPLACE AND GEOMETRIC TRANSFORMS *Walter C. Giffin* 484

GENERATIVE EPISTEMOLOGY OF PROBLEM SOLVING

INTRODUCTION

What can we know? How can we know it? When and where can it be known? These are some of the questions of epistemology. (For a useful overview of epistemological problems, see Wood [57]. The history of this discipline is reviewed in Hamlyn [21].) To these queries, a generative epistemology would add: How can we generate knowledge from a small set of elements and a set of basic relations between these? Is there a "minimal generating set" from which all higher level elements can be derived? (For other approaches to the constructive representation of the world from a small set of elements, see Carnap [15] who uses a single relation of similarity and Spencer-Brown [51] who uses the relation of distinction.) If so, what is it and from whence does it come? The motivation for approaching these questions is quite simple. If an explicit minimal generating set can be specified, and if it can be realized by computing machines, then an "experimental epistemology" (a term introduced by McCulloch [29]) can be developed which would allow the solutions of many problems of epistemology and which would be able to take advantage of the generating power of electronic computing machines. A computer-realized methodology for knowing is a powerful problem-solving tool, as we hope to demonstrate in this article.

This article overviews the types of problems and solutions which underlie the generative approach to the acquisition of knowledge. The foundations for this approach applied to the problem of knowledge integration by computer have evolved over many years through the work of researchers in diversified disciplines throughout the world: Logicians, cyberneticists, mathematicians, computer scientists as well as artists, musicians, philosophers, and even mystics who use the generative methods in their search for truth and knowledge. The future of this method will be determined by the effectiveness of use of the generative and selective potential of the computing machine.

The application of the logistic or method of symbolic logic to axiomatic formulations in the sciences has been masterfully addressed by Carnap [14] and his predecessor Reichenbach [41]. Similarly, Carnap [16] and Reichenbach [42] have also provided excellent introductory pieces in inductive logic. The reader who is interested in multivalued logic is referred to the extensive multilingual bibliography in the book by Rescher [43]. Readers who wish to explore the representation of general machines are referred to the groundbreaking work of Weiner [54] and the didactic treatment of the same material by Ashby [5]. Individuals concerned with the constructive foundations of mathematics might consider the work of Bishop [9] and Markov [33], as well as the unusual approach of Robinson [44]. These are only a representative sampling of currently available sources for the potential generator. A more extensive bibliography is to be found in the author's Rudiments of Generative Epistemology [32].

GENERATIVE EPISTEMOLOGY OF PROBLEM SOLVING

Generative epistemology is a methodology constructed in a language composed of several different object types from which are derived operations and operators. A basic claim of generative epistemology is that all possible modes of knowing are derived from combinations of fundamental activity types: feeling or experiencing; symbolizing or representing; modifying, controlling, changing, or realizing; iterating, repeating, or reflecting—on an experience, a realization, a representation or another reflection; and combining or integrating the component knowledge of the latter four types.

This methodology takes as a starting point a working definition of knowledge and information and, from this, ways for posing and solving problems are constructed. An introductory survey and discussion of a sampling of the methods of generative epistemology are provided for the reader in computer and information science. For a more extensive treatment in both scope and depth of development, the serious reader is referred to the references and bibliography and to the forthcoming book, Rudiments of Generative Epistemology [32], within which will be found an axiomatically developed construction of the language and its application to the arts, technologies, sciences, philosophies, and the integration of these realms of knowledge. In this article a qualitative overview is developed.

The development in this paper proceeds along the following lines: The basic objects, the building blocks of generative epistemology, are introduced, and then interpretations are given and applied to the problem of defining knowledge forms and knowledge types. Next the basic rules for constructing complex object types of generative epistemology are enumerated. This is followed by the generation of complex forms and their interpretation in the context of knowledge and information. The generative epistemology of problem solving is next developed as the primary focus for the audience of computer and information scientists. The classification of problems by enumeration using generative epistemological methods and the development of solutions for types of problems follow. The article concludes with a summary and a prospectus for future work.

FUNDAMENTAL OBJECTS OF GENERATIVE EPISTEMOLOGY

Axiom 1. All objects can be considered as embedded in all other objects.

Definition 1: The specification of an object as embedded in another object is the equivalent of specifying a distribution of one object in terms of the other.

Definition 2: The specification of a distribution of one object in another object is equivalent to the specification of the object, its separation into parts, and the linkages of the parts of the first object to the second object. $\langle O_1/O_2 \rangle$ denotes O_1 embedded in O_2.

Definition 3: The object embedded, O_1, is called the focal object of the distribution, and O_2, the object in which O_1 is embedded, is called the context of the distribution.

Definition 4: If $\langle O_1/O_2 \rangle$ is a distribution and O_3 is any object, then the distribution $\langle O_3/\langle O_1/O_2 \rangle\rangle = O_3/O_1/O_2$ will be called the O_3—O_2 form of O_1, and O_3 will be called the interpreter of the distribution O_1/O_2.

Definition 5: Similarly, if O_1/O_2 is a distribution and O_3 is any object, then the distribution $\langle\langle O_1/O_2\rangle/O_3\rangle = O_1/O_2/O_3$ will be called the O_1—O_3 form of the focal object O_2, and O_3 will be called the context of the distribution O_1/O_2.

Definition 6: In general, the recursion of these forms allows us to define an nth order form for the distribution O_1/O_2:

$$I_n/I_{n-1}/\ldots/I_1/\langle O_1/O_2\rangle/C_1/\ldots/C_{n-1}/C_n$$

In Definitions 3, 4, and 5, the terms focal objects, interpreter, context, and the distributions which connect these elements are related in the construction of a form in which all of these objects are combined into a single object; thus, form = interpreter/focal object/context. Note that the term "context," or contextualizing, restricts the focal object by a projection mapping, and "interpreter" extends the focal object by associating it to a new space.

Context and Interpretation of Knowledge

With these definitions in mind, we proceed intuitively beginning with the notion of uniform differences and nonuniform differences in an object. Uniform and nonuniform differences are represented by distributions (/ is a distribution operator). (Some sources on distribution theory are Schwartz [48], Lighthill [27], and Halperin [20].)

Nonuniform difference in an object will be called the property or attribute of the object. This may arise from a nonuniform difference in the interpreter of the object, in the focal object, in the context, in connections between the object and context (/, O/C), in connections between the interpreter and the object (/, I/O), or in any of these possibilities taken two at a time or as triples. The form of nonuniform difference, that is, ⟨I/discernible difference/C⟩, we call the knowledge of discernible difference. (See Batchelor [7] and Tou [65] for tools for representation of different discernible patterns.) This knowledge object can take forms relative to the differentiating object; these forms are determined by three underlying types of embeddings.

(1) Input type (experiencing): Discernible difference is interpreted, expressed as ⟨specified interpreter/discernible difference⟩. The system is receptive to difference.

(2) Output type (realizing): Discernible difference is contextualized; there is a transmission of discernible difference by the system represented by the string ⟨discernible difference/specified context⟩.

(3) Boundary type (representation): The discernible difference is both received by some system and transmitted by some system. This is a stabilization or storage type: ⟨Specified interpreter/discernible difference⟩ and ⟨discernible difference/specified context⟩.

(4.1) Input form: The input type is contextualized and connected to yield the form ⟨⟨specified interpreter/discernible difference⟩/specified context⟩.

(4.2) Output form: An output type is interpreted and connected to yield the form ⟨specified interpreter/⟨discernible difference/specified context⟩⟩.

(4.3) Boundary form: Both components of the boundary type are connected together to yield the form ⟨specified interpreter/⟨discernible difference⟩/specified context⟩.

(5) Forms of forms: Input forms, output forms, and boundary forms are recursively combined to produce new, integrated forms, defined relative to the discernible difference.

Recursion theory is a mainstay of computer programming as is the simpler notion of iteration. (See Salomaa [47] for some basics on language theory, and Naur [35], McCarthy [28], Wirth [55], and Ollongren [37] for some methods of realizing this relation.)

The nonuniform difference, when embodied in a form, is an object of knowledge, and the specification of the form is the specification of the form of the knowledge. (The role of definitions in any theory is to delineate a starting point which allows the formulation and solution of problems. The superstructure described qualitatively in this article is constructed on an axiomatic base. However, this does not mean that they will not undergo modification—the main motivation in stating them so explicitly is to allow them to be testable by computer and to allow modifications as needed in a scientific way.) Information, or more properly meta-information, is the nonuniform difference form of knowledge; it corresponds to changes in the form of knowledge, while noise corresponds to a uniform difference form of knowledge.

The usual use of information refers to Shannon's syntactical information measures. Shannon considers the situation of change and focuses on the uniform differences (invariances), calling the invariances "information" and the nonuniform differences (variable elements) "noise." In our formulation, we consider a change as meta-information and we note that change can be decomposed into two components—the invariant and variable components. By putting a measure on the invariant component, we can derive Shannon's measure.

Information may arise from any combination of changes in the component parts of a knowledge form, represented as $K_f = I/O/C$. Although in this article we are constraining the discussion to qualitative distinctions, we could also apply the interpreter of quantitative distinctions to the discussion of forms. Use of measures yields quantities of these structures, and formulations such as Shannon's information measure can be derived as special cases.

Specification can be qualitative or quantitative. Specification is a basic association between two universes such that one is expressed in terms of the other, and the value of the association in terms of elements of the context space may be a numerical object or a labeled object. The numerical component omitted in this article is developed elsewhere in detail [32].

Knowledge is the specification of form. Information is a change in the specification of form. A problem is an incompleteness in the specification of form. Solving a problem is the completing of the specification of the incomplete form. A solution makes explicit, relative to the interpreter and the context of a form, that which was previously only implicit. A solut on completes the specification of form to produce knowledge.

For example, suppose we wish to study the behavior of a stream. We can focus on the flow of the stream, and notice that the surface of the stream is sometimes rough and sometimes smooth: a measure of turbulence can quantify this dimension of difference in flow. With this measure we can examine two streams and record the respective degrees of turbulence as T_1 and T_2. By comparing the two measures of turbulence by means of a difference operator, $T_1 - T_2$, we compute a differential turbulence T' which in general can be a distribution. We have begun with an object of focus, stream, and defined on this class of objects a distinguishing feature, turbulence, which forms the context in which the stream is examined. We could further qualify this context by specifying the means by which the differential turbulence is detected—by the unaided human eye, by disturbance of a transversal laser beam, or by some other detector system. The context determines the range within which the differential turbulence or disparity of turbulence can be noticed; outside of this discernibility range, no measurements can be made.

We have identified both an object of focus and a context. To complete the form, we require an interpretation which can be constructed by defining a simple predicate called threat of disparity in turbulence, threat (T'). If T' takes on a set of values $V_1 \ldots V_n$ then the distribution of interpretations can be represented as a simple IF ... THEN ... ELSE sequence:

IF $T' = V_i$ THEN the situation is safe
ELSE IF $T' = V_j$ then do not go swimming
ELSE IF $T' = V_k$ then evacuate region

We specify gradations on the threat dimension such as low threat, high threat, medium threat, and conditions on these interpretations. This conditional selection function is represented in the meta-language by the form:

threat/disparity/turbulence/decision rule/context

If instead of single values of disparity in turbulence we are dealing with complex values, we represent this new level of combination as a selection on the set of all possible combinations and we write a polynomial indicating the weights associated with each parameter. If we analyze this new class of combinations, we can often isolate factors which allow the prediction of change of the complex object. For example, the identification of certain small mountain streams and a change of turbulence at the source of the stream can be used to predict change in the valley river, or the reluctance of birds to land may be an indication of a coming earthquake. In the space of five steps we can design a machine for an early warning system which can be updated and calibrated to develop into an effective sensor for flood control.

To synopsize, we (1) specify nonuniform differences; then (2) specify a discernibility relation for the differences and the knowledge of differences; then (3) specify interpreters, contexts, or connections to yield types of knowledge. We (4) connect the types of knowledge to yield forms of knowledge; and (5) combine forms of knowledge to yield integration of knowledge. Steps (1) through (5) give us a procedure for integrating knowledge. This procedure can be replicated to yield the integration of knowledge at multiple levels such as the component-type level, the form level, and the integration level to build multilevel knowledge structures.

Knowledge Types and Forms

Let us reflect on the distinctions made by reconsidering the knowledge types and forms in more familiar vocabulary.

Experiencing. Sensory input, passive reception, and direct impact of one object on another are some relative synonyms for describing the process of experiencing. This modality may be represented as an interaction type (input type) such that there is a left to right asymmetry with respect to the "signal directed toward" relation. The simple experiential mode is a relation between two elements: The "experiencing" element receives a signal. This modality is input only. I/O represents the interpreting of an object which occurs in this mode.

Realizing. Controlling, modifying, and producing change are some relative synonyms for describing the process of realizing. This modality may be represented as an interaction type such that there is a right-left asymmetry with respect to the "signal directed toward" relation. The simple realization mode is characterized by a relation between two objects such that one element, the "realizing" element, transmits a control signal and the other does not. This modality indicates output only, denoted O/C, which indicates that an object is associated with a context.

Representing. Symbolizing, incorporating, and input-outputting are some relative synonyms for describing the process of representing. This modality is characterized by a symmetry in the "signal directed toward" relation such that there is a relation between two elements, and we focus on the signal embedded in a medium wherein a signal is both received and transmitted. An object in this instance is both the context for some interpreter and the interpreter for some context, where these associations are denoted by O/C and I/O. This representation (boundary type) maintains a pair of distinctions.

Reflecting. Iterating, philosophizing, repeating, and interpreting or contextualizing are some relative synonyms for describing the process of reflecting. This mode produces the first type of form in one of three ways: It produces a context for some input type by connecting the input (I/O) to a context C to yield the form $(I/O)/C = \langle I/O/C \rangle$; it produces an interpreter for some output type by connecting the output type (O/C) to an interpreter to yield the form $I/(O/C) = \langle I/O/C \rangle$; it produces a connection which joins the two distinct representation objects, I/O and O/C, to produce a single form, $(I/(O)/C) = \langle I/O/C \rangle$. We can discern that the process of reflecting introduces a grouping called a form by interpreting a realization, by contextualizing an experience, or by connecting representations. In the latter, it leaves invariant the types of both separate components. In all cases, it connects.

Integrating. Unifying, combining, and generalizing are relative synonyms for describing the process of integrating. The integration operator produces the form of the form of knowledge by adding interpreters, contexts, or connections to forms, yielding higher and higher level combinations. If we consider the focal object itself to be knowledge, then successive iterations and combinations yield higher order structures of knowledge. Since we can decompose by contextualizing or constraining and we can compose or extend by interpreting objects in general, then we can certainly perform these transformations on forms for which the focal object is a discernible difference. Thus, by integrating we mean executing processes of both synthesis and analysis.

Methodology of Generative Epistemology—Overview

We have made the distinctions of the previous section in order to be in the position of having an explicit formulation of what it is to know, of how we can know, of what it is that changes knowledge (information), and the relation of different forms of knowledge. A representation language has been outlined which is formatted to allow the combination of forms and to allow the immediate translation of such combinations into the computer languages of both LISP and APL or, for that matter, into any language which allows the concatenation of objects, a criterion which opens the door to assemblers and even to direct machine representation.

The basic elements of generative epistemology can be realized in many common computer languages. Straightforward tutorials are given in References 8, 13, 24, 28, 38, and 55. What generative epistemology provides is an ontology and meta-logic for configuring these tools for application in many domains.

This brings us to the fundamental problem of generative epistemology. Now that we have a concept of what knowledge is, how can we construct sequences which will lead us from one state of knowledge to another? If we identify a problem as a relation between an initial state and a final state, where these states are forms of knowledge, then how do we specify the initializing and terminating states, successor functions, and constraint rules so that we can generate a sequence of knowledge forms which with suitable specifications of the limits of the space can be made to converge to a desired form? How do we generate a sequence of forms which we can call a procedure for the generation of knowledge? Can we go further and specify algorithms? Under what conditions? Let us begin by considering in some detail the relations involved in problems.

PROBLEM FORMS AND SOLUTION FORMS

Definition 7: A problem is a relation between two systems such that one system is the problem solver and the other system is the target system. This relation is indicated as the projection of the target on the space of the problem solver, namely, target system/problem solver.
In this formulation, the problem solver provides the context relative to the target system and the target system is the interpreter relative to the problem solver; the relations could, of course, be modified by substituting the target system for the problem solver and vice versa, but this would also change the problem.

Definition 8: A problem is posed (stated or specified) when partial information only is given about some attribute of the system as discerned by the problem solver.
For example, the equation $3x = 9$ may pose a problem to a budding algebraist; partial information is available in the form of the equation, including the numerical coefficients and the common interpretations associated with the symbols. The attribute to be established is the value of x expressed as a numeral.

A statement of a problem, that is, the representation of the problem in linguistic terms, involves the use of the imperative mood or the interrogative mood. These linguistic forms oblige the problem solver to perform some process of construction or to execute some procedure or sequence of action so that the partial information knowledge form is transformed into a fully specified knowledge form as evaluated by the meta-interpretations and meta-contexts of the problem, and as constrained by the conditions under which the problem is posed.

We can consider both the evaluation of the problem solution and the conditions under which the problem is executed as interpreters and contexts relative to the problem. If we focus our attention on the constraints on the problem solver, then we are considering the form ⟨target state/problem solver/context⟩. When we focus our attention on the evaluation of the results of the problem solution, relative to the target system, then we are concerned with the problem form ⟨interpreter/target system/problem solver⟩. For example, if our reflection induces us to interpret the target system and problem solver, then we may consider the physical realization of the relation as a procedure executed on computer, with the computer in this case being the relative "experiencer" and the ⟨target system/problem solver⟩ relation the relative "realizer."

Returning now to Definition 8, we can represent the posing of a problem in a kinematic model in which partial information and total information about some attribute of a system is represented as positional distance. Thus, if S_p indicates a state of partial information or knowledge and S_t indicates total knowledge, then separation of these points indicates a problem. A problem form results when the problem type is contextualized or interpreted. In this event the problem solver can provide a path between the initial partial information state and the terminal state by either applying interpretive information transformations or by applying contextualizing information transformations or by application of both. This gives us our basic problem forms with which we can construct problem spaces.

Posing and Solving Problems

So far we have bared the structure of problems so that we can describe the process of posing and solving a problem as the following sequence: (1) The problem solver orients to some target system and notices some nonuniform discernible difference (attribute) of the system. (2) The problem solver discriminates at least some partial information available about the target system since the basic input type, target system/problem solver, implies the discerning of some nonuniform difference in the target system. (3) The kind of partial information available to the problem solver, or the kind of information which the problem solver selects, depends on whether the problem input type is constrained to a context to yield the problem form ⟨target system/problem solver/context⟩, expanded with an interpreter to yield the problem form ⟨interpreter/target system/problem solver⟩, or combined by a connector to yield the form ⟨interpreter/⟨target system/problem solver⟩/context⟩.

But a problem may be posed and forever be ignored. (4) We must add a command to activate the problem form, representing this command by the interpreter "specify." This yields the interpreted problem: specify/⟨problem form⟩. (5) Finally, we can compose and decompose forms in many ways, but a solution must be constrained to a sequence beginning with the initial and terminating with the target state. We represent this solution as a sequence of specifications of component problems in the form sequence/⟨specification/⟨problem form⟩⟩.

To reflect a moment on what this formulation means, consider the initial identification of knowledge as the form of ⟨interpreter/discernible difference/context⟩. This is identical in form to a deactivated problem. An activated problem is solved by explicitly specifying information about the problem form which was previously specified only implicitly. The solution deactivates the problem to produce knowledge. Information is a change in the form of knowledge, and this change may be related to the change of difference between the initial partial-information state of the activated problem and the terminal (relatively) total-information state of the deactivated state. Thus we can say that the problem is solved and knowledge is generated by the application of an information-generating transformation which completes the specification of the knowledge form.

In generative epistemology we provide rules for problem solving in which the set of solution strategies is restricted to generative methods; we rely only on the definition of a small set of elements and a set of basic relations, the elements of which have already been introduced in the concept of form. If we can do this, we will know where our knowledge comes from, and we will be able to explicitly state procedures for realization by computer. We must now consider some specific methods of completing the specification of form by generation using the primitive notions of activation and deactivation.

Generative Methods Using Problem Polynomials

The central problem of generative epistemology is the provision of generative methods for the specification of knowledge. For this process we require methods for representing knowledge types, knowledge forms, and combinations of knowledge forms. In mathematics, number structures are represented by sequences (ordered sets of numbers), by functions (relations), and by polynomials which represent connections of forms and representations of forms. (A good introduction to the tools of linear algebra is Shilov [50], and to algebra in general, MacLane and Birkoff [30]. Realization tools for algebra are discussed by Iverson [24] and Pakin [38].) For example,

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 x^0$$

represents a given distribution (quantity or type) which is formed from combination of a single quantity x with itself: $ax = x + x + \cdots + x$, replicated to some number of times, expressed as a term $a_i x$ and the operation of multiplication, $x^i = x \cdot x \cdot \cdots \cdot x$, expressed as a power. The inner product for the multiplication example is represented as $\cdot(x, x) = x^2$. This general operator, the inner product, can be used to concatenate any elements according to the concatenator used. Scalar product multiplication gives forms (distributions) of the type $a_i x^n$. The general functor, $\oplus$, combines or connects the forms to yield the polynomial form $P(x) = \Sigma a_i x^i$.

Similarly, for sequences or vectors, we have $(a_1, \ldots, a_n)$, where each type is given a place with the first type in place one and so on. This gives rise to a