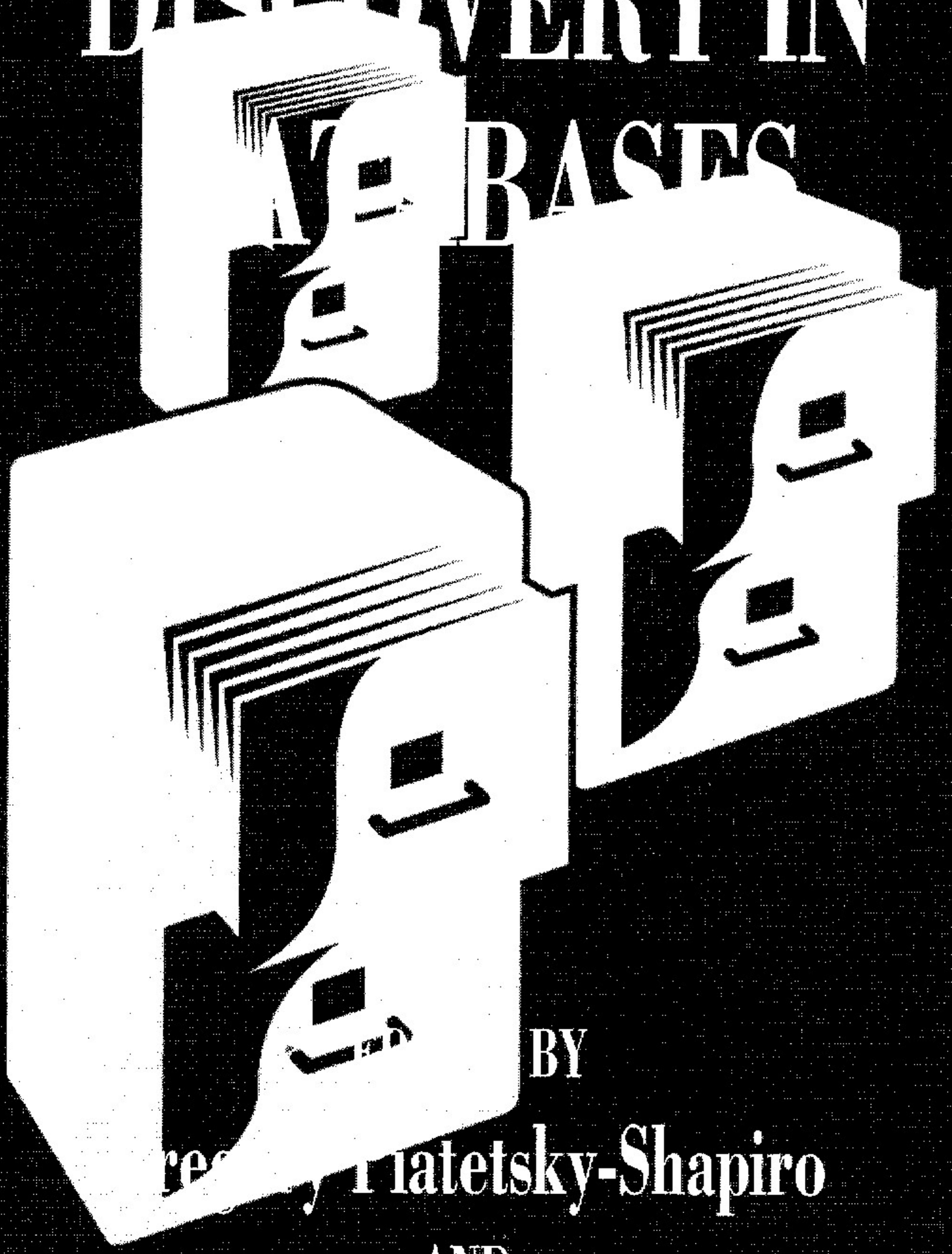
MINING FOR KNOWLEDGE IN DATABASES: GOALS AND GENERAL DESCRIPTION OF THE INLEN SYSTEM

by

K. Kaufman
R. S. Michalski
L. Kerschberg

Chapter in the book, Knowledge Discovery in Databases, G. Piatetski-Shapiro and W. J. Frawley (Eds.), Menlo Park, CA , AAAI Press/The MIT Press, 1991.

KNOWLEDGE DISCOVERY IN DATABASES



BY

Gregory Piatetsky-Shapiro

AND

William J. Frawley

Knowledge Discovery in Databases

AAAI Press / The MIT Press

Menlo Park, California
Cambridge, Massachusetts
London, England

26 Mining for Knowledge in Databases: Goals and General Description of the INLEN System

Kenneth A. Kaufman, Ryszard S. Michalski, and Larry Kerschberg
George Mason University

Abstract

The INLEN system combines database, knowledge base, and machine learning techniques to provide a user with an integrated system of tools for conceptually analyzing data and searching for interesting relationships and regularities in them. Machine learning techniques are used for tasks such as developing general rules from facts, determining differences between groups of facts, creating conceptual classifications of data, selecting the most relevant attributes, determining the most representative examples, and discovering equations governing numeric variables. The equations discovered are accompanied by conditions under which they apply. These techniques are implemented as inference operators that a user can apply to a database or knowledge base to perform a given knowledge extraction function. Examples of three major inference operators are provided, one for learning general rules differentiating between groups of facts, one for creating conceptual classifications of facts, and one for discovering equations characterizing numeric and symbolic data.

26.1 Introduction

This chapter briefly describes the goals and general design of the INLEN system for conceptually analyzing databases and discovering regularities and patterns in them. The name INLEN derives from the terms inference and learning, which represent two major capabilities of the system. INLEN integrates a relational database, a knowledge base, and a number of machine learning and inference capabilities. The latter two enable the system to perform tasks such as creating conceptual descriptions of facts in the database, inventing classifications of data, discovering rules and unknown regularities, and formulating equations together with the conditions of their applicability. We present here a general system design and explain all the basic functions. Major operators, specifically those for determining rules from examples, creating classifications, and discovering equations, are illustrated with examples.

The motivating goal of the INLEN system is to integrate three basic technologies—databases, expert systems, and machine learning and inference—to provide a user with a powerful tool for manipulating both data and knowledge and extracting new or better knowledge from these data and knowledge. INLEN evolved from the QUIN system (query and inference), a combined database management and data analysis environ-

ment (Michalski, Baskin, and Spackman 1982; Michalski and Baskin 1983; Spackman 1983). QUIN was designed both as a stand-alone system and as a subsystem of Advise, a large-scale inference system for designing expert systems (Michalski and Baskin 1983; Michalski, Mozetic, et al. 1987; Baskin and Michalski 1989). In the last few years, new tools have been developed, in particular, more advanced inductive-learning systems, for example, AQ15 (Michalski et al. 1986) and ABACUS-2 (Greene 1988), and expert database systems (Kerschberg 1986, 1987, 1988). These systems have influenced the development of INLEN. INLEN also draws on the experiences with Agassistant, a shell for developing agricultural expert systems (Katz, Fermanian, and Michalski 1986), and Aurora, a general-purpose PC-based expert system shell with learning and discovery capabilities designed by Michalski and Katz (International Intelligent Systems 1988).

However, INLEN is more than just a tool. Its modular architecture enables it to incorporate many discovery tools. INLEN can be viewed as a toolbox, a methodology, or an environment for making all sorts of discoveries in databases. It is especially appropriate to apply INLEN to data systems that are constantly changing or growing; among the systems capabilities are the ability to detect changes over time and explore the ramifications of these changes.

26.2 INLEN System Design

As previously mentioned, INLEN combines database, expert system, and machine learning capabilities to create an environment for analyzing and extracting useful knowledge from a data or knowledge base. It includes ideas from the recently developed expert database technology to combine the storage and access abilities of a database system with the ability to derive well-founded conclusions from a knowledge-based system (Kerschberg 1986, 1987, 1988). INLEN integrates several advanced machine learning capabilities that until now have only existed as separate experimental programs. Many learning systems are capable of but a small subset of what can be learned from factual data. By integrating a variety of these tools, a user will have access to a powerful and versatile system.

The general design of INLEN is shown in figure 26.1. The INLEN system consists of a relational database for storing known facts about a domain and a knowledge base for storing rules, constraints, hierarchies, decision trees, equations accompanied with preconditions, and enabling conditions for performing various actions on the database or knowledge base. The knowledge base not only can contain knowledge about the contents of the database but also metaknowledge for the dynamic upkeep of the knowledge base itself.

The purpose of integrating these capabilities is to provide a user with a set of advanced

tools for searching for, and extracting useful knowledge from, a database; organizing this knowledge from different viewpoints; testing this knowledge on a set of facts; and facilitating its integration within the original knowledge base.

Information in the database consists of relational tables (RTs), and information in the knowledge base consists of units called knowledge segments (KS). A KS can be simple or compound. Simple KSs include rule sets, equations, networks, and hierarchies. Compound KSs consist of combinations of any of these elements or combinations of simple KSs and RTs. The latter form can be used, for example, to represent a clustering that consists of groups of objects (represented as an RT) and the associated descriptions of the groups (represented as rules). Another example of such a representation is a relational table with a set of constraints on, and relationships among, its attributes. These constraints and relationships are represented as rules. Compound KSs also consist of directory tables that specify the locations of their component parts in the knowledge base or, in the case of RT components, in the database.

A justification for such knowledge types is that they correspond to natural forms of representing human knowledge, especially technical knowledge. Also, by distinguishing between these different forms of knowledge and selecting appropriate data structures to represent them, we can achieve greater efficiency in storing and manipulating such structures. Meanwhile, the KS architecture allows for an object-oriented structure in which the user need not be overly concerned about the form taken by a piece of knowledge.

INLEN employs three sets of operators: *data management operators* (DMOs), *knowledge management operators* (KMOs), and *knowledge generation operators* (KGOs). DMOs are standard operators for accessing, retrieving, and manually altering the information in the database. Thus, they operate on RTs. KMOs perform analogous tasks on the knowledge base in situations in which manual input, access, or adjustments are required. The knowledge generation operators interact with both the database and the knowledge base. These operators evoke various situations in which manual input, access, and adjustments are required. KGOs take input from both the database and the knowledge base. These operators invoke various machine learning programs to perform tasks such as developing general rules from facts, determining differences between groups of facts, creating conceptual classifications of data, selecting the most relevant attributes, determining the most representative examples, and discovering equations governing numeric variables. The results of KGOs are stored as KSs. Examples of the performance of a few basic knowledge generation operators are given in An Illustration of Selected Knowledge Generation Operators: Cluster, Diff, and Diseq. A brief description of DMOs, KMOs, and KGOs follows.

26.2.1 Data Management Operators

DMOs form a standard set of relational database operations for the purpose of manipulating the system's collection of facts:

Create generates a new relational table. It takes an attribute list as an argument.

Append adds a new tuple (row) to a relational table.

Change alters some or all of the values in some or all of the tuples of a table.

Delete removes rows or columns from a table, as specified, respectively, by Select or Project operations. Alternatively, entire tables can be removed from the system.

Select retrieves a relational table from a database and returns the complete table or part of it. The part represents the subset of its rows that satisfy criteria specified in the arguments of the operator. Project reduces a table by removing columns. Columns that are kept correspond to attributes specified in the arguments of the operator.

Join creates a relational table combining the columns of two tables. The rows are the subset of the rows of the Cartesian product of the two tables whose attributes satisfy criteria provided by the user.

Union, performed on two tables with the same set of attributes, returns the set of tuples (rows) that appear in either of the two tables.

Intersect, performed on two tables with the same set of attributes, returns the set of tuples that appear in both of the input tables.

26.2.2 Knowledge Management Operators

KMOs are used to create, manipulate, and modify INLEN's knowledge base, thereby allowing the knowledge base to be handled in a manner analogous to handling a database. Knowledge can take the form of simple or compound KSs. Consequently, most of KMOs shown in figure 26.1 are generalized for any of these forms. Unless otherwise specified, they should be thought of as operating on any KS; that is, they can operate on rules, equations, hierarchies, and so on.

Diverse representations of knowledge can be culled from the same database and, therefore, will represent distinct viewpoints obtained using the knowledge generation operators. For example, a dynamic system whose behavior is governed by a set of differential equations could have its time series input-output behavior represented as a relation consisting of all measurable input-output variables. Each tuple would consist of the input-output variable value at some time. KGOs could be used to create knowledge viewpoints such as functional and multivalued dependencies from relational database theory, a set of decision rules, a causal and temporal semantic network, and so on. Each of these viewpoints is valid and should be managed by KMOs.

Expert database tools and techniques can be used to manage the evolution of the com-

bined knowledge database by incorporating knowledge discovered in the database. The arrow in figure 26.1 linking the database and the knowledge base components represents such an interaction.

KMOs listed in the following are depicted as analogues of INLEN's data management operators. Without intensive testing of the system in different domains, one cannot tell how useful these operators are, but they represent our first approximation based on the analogy with DMOs. Further research might lead to the development of other operators and also other knowledge representations, including the likely use of a more object-oriented approach in which one data representation is replaced by an active link with the concept of a formula, a rule set, or some other representation.

Under the current design, these are KMOs and their functions:

Create is used to generate a new KS with a structure and set of attributes specified by the user. KS will be empty until knowledge is added using either an Append operator or one of the knowledge generation operators.

Append is used for the manual addition of new knowledge to KS.

Change is used for the manual alteration of part of one or more items in KS.

Delete is used to remove selected portions of KS from the knowledge base. Alternatively, an entire KS can be erased by giving no qualifying conditions to the operator.

Select is used to retrieve KS from the knowledge base (and from the database in the case of component RTs). Criteria can be provided to return only selected items (such as rules, subtrees, rows in tables, and so on) in this KS.

Project is used to return a subset of a compound KS that ignores entire components (for example, rule sets, decision trees, columns of tables) of KS. The items specified in the operator's arguments will be included.

Join is used to combine a pair of simple KSs or components of compound KSs. For example, a set of rules and a data table can be united into a compound KS, or two rule sets can be combined by finding conditions in the first rule set that are satisfied by decisions in the second rule set. Rules can then be expanded by replacing the matching conditions in the first rule set with the conditions leading to the corresponding decisions in the second rule set.

Union is applied to two or more KSs of the same type. It generates a list of the elements present at least once in any of the segments.

Intersect is applied to two or more KSs of the same type. It generates a list of the elements present at least once in each of the segments.

26.2.3 Knowledge Generation Operators

KGOs perform complex inferences (often approximating NP-complete tasks) on KSs to create new knowledge. It should be noted that KGOs also consist of primitives (such as

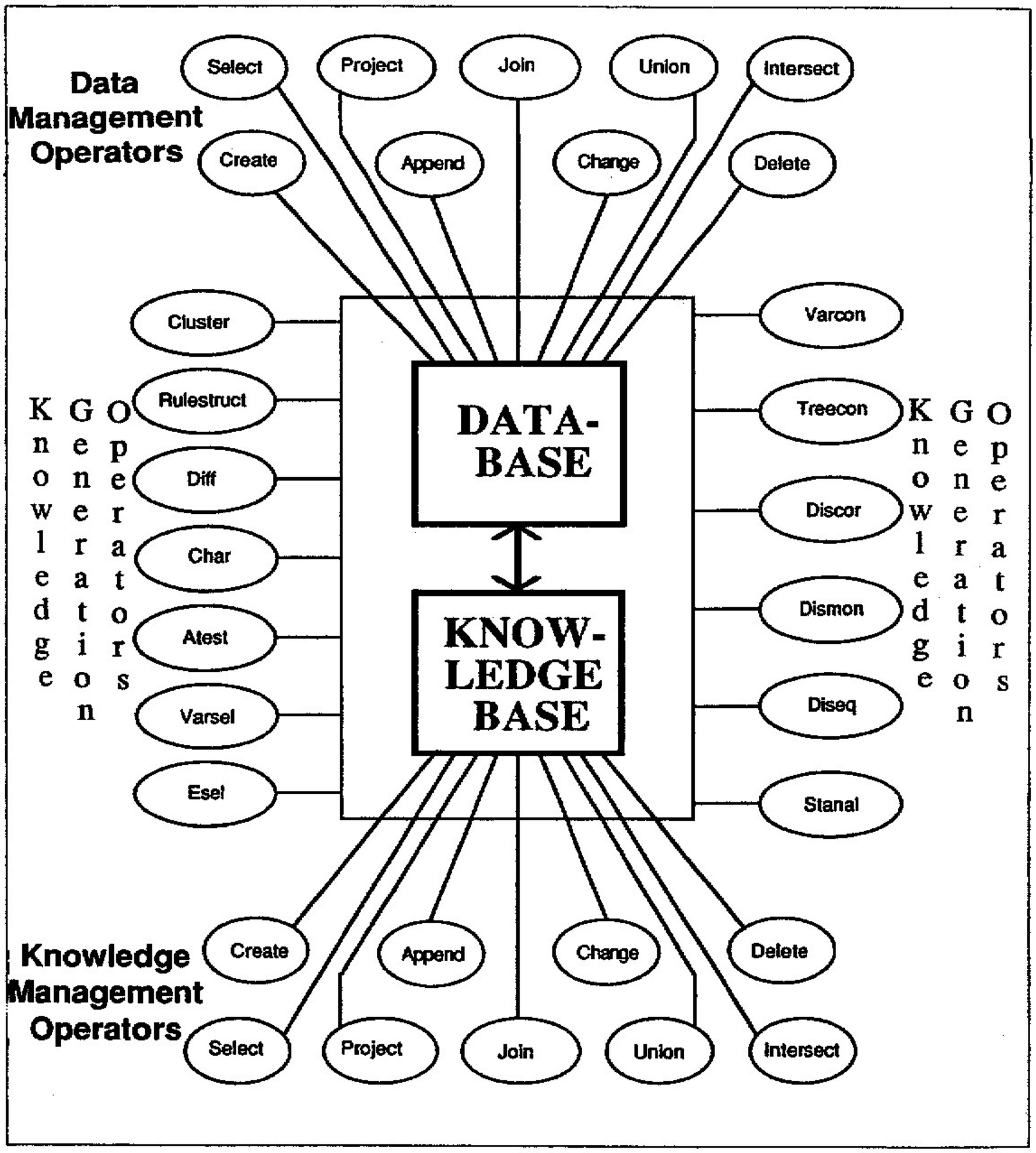


Figure 26.1
A Functional Diagram of INLEN.

save and retrieve) to facilitate access to the structures they generate. These structures will generally be compound KSs that include tables in the knowledge base that locate their other components.

Many of these operators work with or generate rules. Rules in INLEN consist of a decision part implied by a condition part. The decision part consists of a conjunction of one or more statements or actions, and the condition part consists of a disjunction of conjunctions, each consisting of one or more elementary conditions (for examples, see tables 26.1 and 26.2).

Under the current design, these are the basic KGOs employed by INLEN:

Cluster performs conceptual clustering of tuples in a relational table to create logical groupings of objects or events represented by the tuples. It also determines a set of rules characterizing the created groups. Specifically, the operator divides rows of a relational table into two or more groups and returns KS consisting of a relational table similar to the input table and containing additional information indicating the groups and a rule set characterizing the individual groups. An example of this operator is given in the next section. User-defined parameters can influence the creation of the groups. Detailed descriptions of the conceptual clustering algorithm that performs this operator are in Michalski, Stepp, and Diday (1981) and Stepp (1983, 1984).

Rulestruct also performs conceptual clustering but applies it to a rule set rather than a relational table. A compound KS is returned consisting of the original rule set with grouping information plus a new rule set to explain the grouping.

Diff (differentiate) takes two or more classes of objects (each object represented as a tuple in a relational table) and induces general rules characterizing the differences between the classes. The output KS consists of the rule set created by the operator and the object classes represented by RTs. The AQ program that executes this operator is described in Michalski and Larson (1983). The rules produced are called *discriminant descriptions*; that is, they specify sufficient conditions for distinguishing one class of objects from the other class(es).

Char (characterize) determines descriptions characterizing a class of objects. This operator also falls into the domain covered by the AQ program. Here, the emphasis is on finding characteristic rules describing all examples of a class of objects, without concern about the differences between this class and other classes. Output includes the initial class plus the generated descriptions.

Atest tests a set of decision rules for consistency and completeness on a set of examples (specified in a relational table). Consistency implies that no event in the example space is covered by two different rules. Completeness refers to the condition that every possible example will be covered by the conditions applying to at least one rule. The output KS consists of the input rules, example sets, and a relational table containing Atest's

analysis. Atest is described in detail in Reinke (1984).

Varsel determines attributes in a relational table that are most relevant for differentiating between various classes of objects. Output consists of a rule describing the selection of the variables given the input classes and the subtable generated by projecting on the chosen variables. By keeping only the most relevant attributes in the object (example) descriptions, one can significantly reduce the computation time required by the Cluster or Diff operators (Baim 1982).

Esel determines the examples (objects) that are most representative for given classes. Promising examples are returned as output with a rule specifying the input classes and the chosen examples, and other examples are rejected (Michalski and Larson 1978; Cramm 1983).

Varcon applies mathematical operators specified in its argument to combine variables into useful composites. The output KS will consist of the new composite variables and a rule specifying the original table, the mathematical operators, and the created variables. For example, Varcon can be used if the sum or product of two variables might be more useful than either individual value (Davis 1981).

Treecon takes a set of rules or decision examples and organizes them into a decision tree, which can be a more efficient way for storing or using this knowledge (Michalski 1978; Layman 1979).

Discor discovers correlations between the values of attributes in a set of examples. It is implemented as a standard statistical operation of correlation and returns a table of its results.

Dismon seeks out monotonic relations between attributes in a set of examples and in doing so can discover an interesting relationship within the data. It is an operator that is used in the Diseq operator.

Diseq discovers equations that describe numeric data in a set of examples and formulates conditions for applying these equations. Diseq returns a set of equations and the rules that determine when they apply. It is based on the ABACUS-2 system for integrated qualitative and quantitative discovery (Falkenhainer and Michalski 1986; Greene 1988). ABACUS-2 is related to programs such as BACON (Langley, Bradshaw, and Simon 1983), FAHRENHEIT (Żytkow 1987), and COPER (Kokar 1986).

Stanal performs a statistical analysis of the data to determine its various statistical properties.

Table 26.1
An Example of the Cluster Operator.

	Input					Output	
Microcomputer	Display	RAM	ROM	Processor	No_Keys	2-Group	3-Group
Apple II	Color_TV	48K	10K	6502	52	1	1
Atari 800	Color_TV	48K	10K	6502	57-63	1	1
Comm. VIC20	Color_TV	32K	11-16K	6502A	64-73	1	2
Exidi Sorceror	B/W_TV	48K	4K	Z80	57-63	1	2
Zenith 118	Built_in	64K	1K	8080A	64-73	2	3
Zenith 1189	Built_in	64K	8K	Z80	64-73	2	3
HP 85	Built_in	32K	80K	HP	92	1	2
Horizon	Terminal	64K	8K	Z80	57-63	1	2
Challenger	B/W_TV	32K	10K	6502	53-56	1	1
O-S 11 Series	B/W_TV	48K	10K	6502C	53-56	1	2
TRS-80 I	B/W_TV	48K	12K	Z80	53-56	1	1
TRS-80 III	Built_in	48K	14K	Z80	64-73	1	1

Two-Group Clustering:
[Group 1] $\Leftarrow$ [RAM = 16K..48K] or [No_Keys $\leq$ 63]
[Group 2] $\Leftarrow$ [RAM = 64K] & [No_Keys > 63]

The Cluster operator takes as the input the relational table, marked Input, and a parameter requiring it to partition the rows in the table into two- and then three-group clusterings. The two rightmost columns show the partitions generated. Cluster also generates rules describing the groups stored in the knowledge base

Three-Group Clustering:
[Group 1] $\Leftarrow$ [Processor = 6502 v 8080A v Z80] & [ROM = 10K..14K]
[Group 2] $\Leftarrow$ [Processor = 6502A v 6502C v HP] or [ROm = 1K..8K] & [Display $\neq$ Built_in]
[Group 3] $\Leftarrow$ [Processor = 6502 v 8080A v Z80] & [ROM = 1K..8K] & [Display = Built_in]

26.3 An Illustration of Selected Knowledge Generation Operators: Cluster, Diff, Diseq

This section gives examples of how some basic KGOs work, specifically, the Cluster, Diff, and Diseq operators.

26.3.1 Cluster

Cluster is capable of creating groupings of objects or events and, when used recursively, can generate an entire taxonomy. Unlike traditional clustering methods, Cluster also returns the rules that describe its grouping. The presented example is based on the results described in Michalski and Stepp (1983), which involves creating a classification of microcomputers. Variables considered include the type of processor, the amount of random-access memory (RAM), the read-only memory (ROM) size, the type of display, and the number of keys on the keyboard. Dividing the examples into two groups, the system grouped them according to RAM size and keyboard; clustering into three groups was based on the processor type, ROM size, and the display type. Table 26.1 presents the original data and the classifications generated by the Cluster operator. The input to Cluster was a table of the characteristics of the microcomputers, and the output consisted of a table with new columns indicating the groups of the objects along with rules characterizing the groups.

Table 26.2
An Example of the Diff Operator.

Microcomputer	Display	RAM	ROM	Processor	No_Keys	2-Group	3-Group
Apple II	Color_TV	48K	10K	6502	52	1	1
Atari 800	Color_TV	48K	10K	6502	57-63	1	1
Comm. VIC20	Color_TV	32K	11-16K	6502A	64-73	1	2
Exidi Sorceror	B/W_TV	48K	4K	Z80	57-63	1	2
Zenith 118	Built_in	64K	1K	8080A	64-73	2	3
Zenith 1189	Built_in	64K	8K	Z80	64-73	2	3
HP 85	Built_in	32K	80K	HP	92	1	2
Horizon	Terminal	64K	8K	Z80	57-63	1	2
Challenger	B/W_TV	32K	10K	6502	53-56	1	1
O-S 11 Series	B/W_TV	48K	10K	6502C	53-56	1	2
TRS-80 I	B/W_TV	48K	12K	Z80	53-56	1	1
TRS-80 III	Built_in	48K	14K	Z80	64-73	1	1

Diff takes as input a relational table in which the last column indicates group (class) membership. In this example, the Diff operator tries to rediscover the rules, invented by Cluster, from the examples of groups.

Rediscovered rules for Two-Group Differentiation:

[Group 1] $\Leftarrow$ [Display $\neq$ Built_in] or [ROM $\geq$ 14K]
[Group 2] $\Leftarrow$ [RAM = 64K] & [No_Keys = 64-73]

Rediscovered Rules for Three-Group Differentiation:

[Group 1] $\Leftarrow$ [Processor = Z80 v 6502] & [ROM = 10K..14K]
[Group 2] $\Leftarrow$ [Processor = 6502C v 6502A v HP] or [ROM = 4K..8K] & [Display = B/W_TV v Term.]
[Group 3] $\Leftarrow$ [ROM = 1K..8K] & [Display = Built_in]

These rules were generated by Diff directly from examples. They are similar but not identical to the rules originally created by Cluster. They provide an alternative, logically consistent, characterization of individual groups.

26.3.2 Diff

The Diff operator is based on the AQ inductive-learning method that has been effectively used for many rule-learning tasks in areas such as medicine, agriculture, physics, computer vision, and chess. One recent application for diagnosing potential breast cancers, given a few training examples, is described in Michalski, Iwanska, et al. (1986). The rules generated performed well on new cases of the disease. An application of the Diff system to concisely describe the groups created by the Cluster operator (table 26.1) is shown in table 26.2. The groups of examples are given as input, and Diff creates rules that describe the differences between these groups. Note that the found rules are a little simpler than the descriptions produced by Cluster (a redundant condition specifying the processor type in the third group of the three-grouping cluster was removed). Diff often produces a significantly simpler description.

Although this example shows an application of Diff to create the discriminant rules for groups of examples, the AQ algorithm that it employs can also be used to determine characteristic rules that describe classes of events (Michalski 1983). In the INLEN system, this function is represented by the Char operator. In case of large example sets, there can be large differences between characteristic and discriminant rules.

Table 26.3
The Diseq Operator Formulates Stoke's Law.

Substance	Radius (m)	Mass (kg)	Height (m)	Time (s)	Velocity (m/s)
Vacuum	0.05	1	6	0.1	0.98453
Vacuum	0.05	2	2	0.4	3.93812
Vacuum	0.10	1	3	0.5	2.95359
Vacuum	0.10	2	7	0.1	0.98453
Glycerol	0.05	1	5	0.1	19.11200
Glycerol	0.05	2	8	0.3	38.22400
Glycerol	0.10	1	6	0.5	9.55600
Glycerol	0.10	2	7	0.2	19.11200
CastorOil	0.05	1	9	0.4	14.67200
CastorOil	0.05	2	3	0.1	29.34400
CastorOil	0.10	1	5	0.3	7.33600
CastorOil	0.10	2	8	0.5	14.67200

Diseq searches for relationships among the data objects. It discovers that equations for the ball's velocity exist, but they depend on the medium through which the ball is falling.
Here are the rules Diseq discovered:
If [Substance = Vacuum] then $v = 9.8175 * t$
If [Substance = Glycerol] then $v * r = 0.9556 * m$
If [Substance = CastorOil] then $v * r = 0.7336 * m$,
where v = velocity, r = radius, t = time, and m = mass.

26.3.3 Diseq

The Diseq operator is based on the Abacus-2 discovery system that is described in Greene (1988). The operator is capable of learning equations that fit a set of tabular data. It is also capable of subdividing a set of examples into subsets in which different rules apply and coping with noisy data. It specifies conditions under which different rules apply. Abacus-2 expands the capabilities of the earlier system Abacus (Falkenhainer and Michalski 1986) and can discover more complex regularities.

The Abacus programs have formulated equations characterizing a number of different empirical data, for example, data specifying planetary motion, the distances between atoms in a molecule, and Stoke's law of falling bodies. Stoke's law specifies the velocity of an object falling through different media and is presented in table 26.3. As shown in the table, the velocity of an object falling through a fluid is governed by an equation involving different variables from those found in the equation describing the velocity of an object falling through a vacuum. Diseq was able to find the equations for both cases.

26.4 Conclusion

INLEN is a large-scale integrated system capable of performing a wide variety of complex inferential operations on data to discover interesting regularities in them. These regularities can be detected in qualitative data and quantitative data as well as in the knowledge base itself. In addition, INLEN provides functions that facilitate manipulation of both the data and the knowledge base.

A major novelty of INLEN is that it integrates a variety of knowledge generation oper-

ators that permit a user to search for various kinds of relationships and regularities in the data. To achieve such an integration, the concept of KS was introduced. The KS stands for a variety of knowledge representations, such as rules, networks, and equations, each possibly associated with a relational table in the database (as in the case of a set of constraints), or for any combination of such basic KSs.

Because INLEN serves to collect learning and discovery systems as operators, it is more of a methodology than a simple tool. Because it incorporates many diverse knowledge generation operators, INLEN carries the possibility of extending the limits of discovery systems' capabilities.

Many of INLEN's modules have already been implemented as stand-alone systems or parts of larger units. Other tools and the general integrated interface are under development. Future work involves bringing these systems together and completing the control system to facilitate access to the systems in the form of simple, uniform commands.

Acknowledgments

The authors thank Pawel Stefanski, Jianping Zhang, and Jan Zytkow for their comments and criticism. They are also grateful to Peter Aiken, Kathleen Byrd, and Joyce Ralston for their assistance in preparing earlier versions of the manuscript.

This research was done in the Artificial Intelligence Center at George Mason University. The activities of the center are supported in part by the Defense Advanced Research Projects Agency under a grant administered by the Office of Naval Research N00014-87-K-0874 and in part by the Office of Naval Research, grants N00014-88-K-0226 and N00014-88-K-0397.

References

Baim, P. W. 1982. The PROMISE Method for Selecting Most Relevant Attributes for Inductive Learning Systems, UIUCDCS-F-82-898, Dept. of Computer Science, Univ. of Illinois.

Baskin A. B., and Michalski, R. S. 1989. An Integrated Approach to the Construction of Knowledge-Based Systems: Experiences with ADVISE and Related Programs. In *Topics in Expert System Design*, eds. G. Guida and C. Tasso, 111–143. Amsterdam: Elsevier.

Cramm, S. A. 1983. ESEL/2: A Program for Selecting the Most Representative Training Events for Inductive Learning, UIUCDCS-F-83-901, Dept. of Computer Science, Univ. of Illinois.

Davis, J. H. 1981. CONVART: A Program for Constructive Induction on Time Dependent Data, Master's thesis, Dept. of Computer Science, Univ. of Illinois.

Falkenhainer, B., and Michalski, R. S. 1986. Integrating Quantitative and Qualitative Discovery: The ABACUS System, UIUCDCS-F-86-967, Dept. of Computer Science, Univ. of Illinois.

Greene, G. 1988. Quantitative Discovery: Using Dependencies to Discover Non-Linear Terms, Master's thesis, Dept. of Computer Science, Univ. of Illinois.

International Intelligent Systems, Inc. 1988. User's Guide to AURORA 2.0: A Discovery System, International Intelligent Systems, Inc., Fairfax, Va.

Katz, B.; Fermanian, T. W.; and Michalski, R. S. 1986. AgAssistant: An Experimental Expert System Builder for Agricultural Applications, UIUCDCS-F-87-978, Dept. of Computer Science, Univ. of Illinois.

Kerschberg, L., ed. 1988. *Expert Database Systems: Proceedings from the Second International Conference*, Fairfax, Va.: George Mason Univ.

Kerschberg, L., ed. 1987. *Expert Database Systems: Proceedings from the First International Conference*. Menlo Park, Calif.: Benjamin Cummings.

Kerschberg, L., ed. 1986. *Expert Database Systems: Proceedings from the First International Workshop*. Menlo Park, Calif.: Benjamin Cummings.

Kokar, M. M. 1986. Coper: A Methodology for Learning Invariant Functional Descriptions. In *Machine Learning: A Guide to Current Research*, eds. R. Michalski, T. Mitchell, and J. Carbonell, 151–154. Boston, Mass.: Kluwer.

Langley, P.; Bradshaw, G. L.; and Simon, H. A. 1983. Rediscovering Chemistry with the BACON System. In *Machine Learning: An Artificial Intelligence Approach*, Volume 1, eds. R. Michalski, T. Mitchell, J. Carbonell, 221–240. San Mateo, Calif.: Morgan Kaufmann.

Layman, T. C. 1979. A Pascal Program to Convert Extended Entry Decision Tables into Optimal Decision Trees, Internal Report, Dept. of Computer Science, Univ. of Illinois.

Michalski, R. S. 1983. Theory and Methodology of Inductive Learning. In *Machine Learning: An Artificial Intelligence Approach*, volume 1, eds. R. Michalski, T. Mitchell, and J. Carbonell, 83–134. San Mateo, Calif.: Morgan Kaufmann.

Michalski, R. S. 1978. Designing Extended Entry Decision Tables and Optimal Decision Trees Using Decision Diagrams, UIUCDCS-R-78-898, Dept. of Computer Science, Univ. of Illinois.

Michalski, R. S., and Baskin, A. B. 1983. Integrating Multiple Knowledge Representations and Learning Capabilities in an Expert System: The ADVISE System. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 256–258. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Michalski, R. S., and Larson, J. B. 1983. Incremental Generation of VL_1 Hypotheses: The Underlying Methodology and the Description of the Program AQ11, UIUCDCS-F-83-905, Dept. of Computer Science, Univ. of Illinois.

Michalski, R. S., and Larson, J. B. 1978. Selection of Most Representative Training Examples and Incremental Generation of VL1 Hypotheses: The Underlying Methodology and the Description of Programs ESEL and AQ11, 867, Dept. of Computer Science, Univ. of Illinois.

Michalski, R. S., and Stepp, R. E. 1983. Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(4):396–410.

Michalski, R. S.; Baskin, A. B.; and Spackman, K. A. 1982. A Logic-Based Approach to Conceptual Database Analysis. In Proceedings of the Sixth Annual Symposium on Computer

Applications in Medical Care (SCAMC- 6), 792-796. Washington, D.C.: George Washington Univ. Medical Center.

Michalski, R. S.; Stepp, R. E.; and Diday, E. 1981. A Recent Advance in Data Analysis: Clustering Objects into Classes Characterized by Conjunctive Concepts. In *Progress in Pattern Recognition*, volume 1, eds. L. N. Kanall and A. Rosenfeld, 33–56. New York: North Holland.

Michalski, R. S.; Baskin, A. B.; Uhrik, C.; and Channic, T. 1987. The ADVISE.1 Meta-Expert System: The General Design and a Technical Description, UIUCDCS-F-87-962, Dept. of Computer Science, Univ. of Illinois.

Michalski, R. S.; Iwanska, L.; Chen, K.; Ko, H.; and Haddawy, P. 1986. Machine Learning and Inference: An Overview of Programs and Examples of Their Performance, Artificial Intelligence Laboratory, Dept. of Computer Science, Univ. of Illinois.

Michalski, R. S.; Mozetic, I.; Hong, J.; and Lavrac, N. 1986. The AQ15 Inductive Learning System: An Overview and Experiments. UIUCDCS-R-86-1260, Dept. of Computer Science, Univ. of Illinois.

Reinke, R. E. 1984. Knowledge Acquisition and Refinement Tools for the ADVISE Meta-Expert System, Master's thesis, Dept. of Computer Science, Univ. of Illinois.

Spackman, K. A. 1983. QUIN: Integration of Inferential Operators within a Relational Database, ISG 83-13, UIUCDCS-F-83-917, Master's thesis, Dept. of Computer Science, Univ. of Illinois.

Stepp, R. E. 1984. Conjunctive Conceptual Clustering: A Methodology and Experimentation, Ph.D. diss., Dept. of Computer Science, Univ. of Illinois.

Stepp, R. E. 1983. A Description and User's Guide forCLUSTER/2, a Program for Conceptual Clustering, Dept. of Computer Science, Univ. of Illinois.

Żytkow, J. M. 1987. Combining Many Searches in the FAHRENHEIT Discovery System. In *Proceedings of the Fourth International Workshop on Machine Learning*, 281–287. San Mateo, Calif.: Morgan Kaufmann.