SEARCHING FOR KNOWLEDGE IN LARGE DATABASES

by

R. S. Michalski
L. Kerschberg
K. Kaufman
J. Ribeiro

Proceedings of the First International Conference on Expert Systems and Development, Cairo, Egypt, April 1992.

Invited paper at the First International Conference
on Expert Systems and Development
Cairo, Egypt, April 19-23, 1992

SEARCHING FOR KNOWLEDGE IN LARGE DATA BASES

R. S. Michalski†,
L. Kerschberg†,
K. A. Kaufman†,
J. S. Ribeiro† *

† Center for Artificial Intelligence
George Mason University
Fairfax, VA 22030

* The Analytic Sciences Corporation
12100 Sunset Hills Road
Reston, VA 22090

Abstract: Among the central tasks in the development of expert systems is the formulation, debugging and implementation of a knowledge base. The knowledge encoded in the knowledge base is usually supplied by experts. There are, however, many application domains in which knowledge required by an expert system has to be extracted from facts collected in a data base. In view of the large sizes and the complexity of contemporary data bases in different areas, such as agriculture, medicine, business, etc., determining useful knowledge from them is becoming an increasingly difficult problem. This paper describes a multistrategy "intelligent assistant" for knowledge discovery in large data bases, called INLEN. The system integrates a database, a knowledge base, and machine learning capabilities within a uniform user-oriented framework. The latter capabilities are incorporated into the system in the form of *knowledge generation operators* (KGOs). These operators can generate diverse kinds of knowledge about the properties and regularities existing in the data. For example, they can hypothesize general diagnostic rules from specific cases of diagnosis, optimize the rules according to problem-dependent criteria, determine differences and similarities among groups of facts, propose new variables, create conceptual classifications, determine equations governing numeric variables and the conditions under which the equations apply, derive statistical properties and use them for qualitative evaluations, etc. The initially implemented system, INLEN-1, is described, and its performance is illustrated by an example.

Keywords: Knowledge discovery, machine learning, databases, multistrategy systems.

Introduction

From the time when large computer storage media became available in the late fifties, there has been an extraordinary growth of computer databases in almost every area of human endeavor. Whether in agriculture, medicine, industry, military, government or science, thousands of databases have been developed to capture information relevant to some particular class of activities.

There has not been, however, corresponding progress in the methods for extracting useful knowledge from these databases. Many programs have been developed to analyze data, but these techniques typically employ various statistical methods, and as such have certain intrinsic limitations. A statistical analysis can detect a correlation between given factors, but does not produce a conceptual explanation why such a correlation exists, nor does it produce any qualitative description characterizing this correlation. A statistical technique can determine a central tendency and variability of given properties, but it cannot explain them in terms of causal dependencies. It can fit an equation to a set of datapoints, but cannot qualitatively characterize the applicability conditions for this equation, nor modify or create new variables and involve them in the equation. A numerical taxonomy technique can create a classification of entities and determine a numerical similarity among the entities in the same or different classes, but

it will not create a qualitative explanation of the classes created. Attributes that define a similarity and the measures of similarity involved must be given in advance. In short, the techniques mentioned above require that an interpretation of the findings, i.e., a "conceptual" analysis of data, be performed by a human analyst. As the quantity of available data increases, the complexity of such an analysis may outstrip human capabilities.

The goal of this research is to overcome some of these limitations by employing modern techniques of machine learning and discovery. These new techniques open new possiblities for deriving useful knowledge directly from data, and thus provide new tools for the development of expert systems. Early research on the application of machine learning to the development of diagnostic rules, and to building agricultural expert systems have shown a great potential of such an approach. For example, Chilausky, Jacobsen & Michalski (1976) have succefully demonstrated the applicability of machine learning techniques to learning diagnostic rules for soybean diseases from exmples of diseases. Subsequently, this method was used in the first expert system with learning capabilities (Michalski & Chilausky, 1980); Michalski et al.; 1982a; Uhrik, 1982; Michalski et al. 1983; Reinke, 1983). Machine learning techniques found also application in other agricultural domains, for example, for predicing black cutworm damage in corn (Boulanger, 1983), for ground development (Newkirk, 1985), and for weed identification (Fermanian, Michalski and Katz, 1985). Subsequently, these techniques have been incoporated in an agricutural expert system development tool, called AgAssistant (Katz, Fermanian and Michalski, 1987; and Fermanian and Michalski, 1992). It may be relevant to mention that these techniques also have also applications to medicine (e.g., Mozetic, 1985; Bratko, Mozetic and Lavrac, 1988).

This paper describes the architecture and initial implementation of a large-scale system, called INLEN, for conceptually analyzing databases and discovering regularities in them. The name INLEN derives from inference and learning, which represent two major capabilities of the system. INLEN integrates a relational database, a knowledge base and a variety of machine learning programs. The latter ones are implemented in the form of *knowledge generation operators* that create new "nuggets" of data and/or knowledge from given data and knowledge. The operators employ all basic forms of inference—deduction, induction and analogy. These capabilities allow a user to explore the data using a variety of strategies, and therefore INLEN can be viewed as a *multistrategy* data exploration system. In addition to knowledge generation operators, INLEN also includes methods for visualizing different types of data and knowledge, and for generating explanations for the discovered relationships.

The basic approach is to build an "intelligent assistant" that can amplify the effectiveness of experts by generating important findings and explanations of them partially on its own. The system includes criteria and heuristics characterizing the type of information or classes of patterns that might be important to a user. It also provides a variety of advanced plausible reasoning techniques for implementing a search for such patterns. The role of such an intelligent assistant is to search for all kinds of qualitative and/or quantitative patterns, and to notify an analyst about the patterns viewed as important. These patterns would be formulated by applying methods of symbolic concept learning. Experiments with some existing machine learning programs have shown that these programs can find unexpectedly simple patterns that are difficult for people to recognize (e.g., Michalski,1983), or discover regularities that would be hard to formulate without an aid of a program (e.g., Falkenhainer & Michalski, 1990). Some patterns that learning programs find may turn out to be irrelevant, but some of them may turn out to be truly important. When a database is large, it may be difficult for an analyst to find patterns due to the sheer volume of data. It can also help to avoid the possible human error of overlooking something of note in the data.

INLEN's approach is to build a synergistic system that allows a human expert and a computer tool to perform the tasks that each party is better suited for. Some patterns are more easily detectable by a machine than by humans; others are obvious to the human eye, but difficult to notice by today's discovery systems. Data and knowledge management functions, searches through large data sets, consistency checking and discovery of certain classes of patterns are relatively easy to perform by a learning and discovery system. On the other hand, defining criteria for judging what is important and what is not, making decisions about what data to process, and evaluating findings from the viewpoint of human utility are easier for a human expert.

Systems able to extract useful knowledge from large databases can be useful in many fields. Their major application is for automated knowledge aquisition for expert systems in such areas as classification decision making, resource allocation and management, business transactions, agriculture, medicine, chemistry, physics, economics, demographics, global change, scheduling, planning, etc. Workers in all of these areas have contact with large amounts of data, much of which is, or can be, stored in databases. Current methods for knowledge acquisition are complex, time-consuming and error-prone (e.g., Boose & Gaines, 1989; Boose, Gaines & Ganascia, 1989; Boose, Gaines & Linster, 1988). Most of the difficulties come from the fact that the system lacks the capability of self-improving its knowledge through experience, i.e. lacks learning abilities.

The underlying data and knowledge representation in INLEN is *knowledge segments*, which link relational tables with rules, equations and/or hierarchies. The knowledge segment is a flexible structure for storing background or discovered knowledge about the facts in the database. Its format is designed to facilitate interaction with other data and/or knowledge, and to facilitate the user's understanding of the concepts stored within.

INLEN evolved from the QUIN system (**Q**uery and **I**nference), a combined database management and data analysis environment (Michalski & Baskin,1983; Michalski, Baskin & Spackman, 1982; 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 (Baskin & Michalski, 1989; Michalski & Baskin, 1983; Michalski et al. 1987). In the last few years, new tools have been developed; in particular, more advanced inductive learning systems, e.g., AQ 15 (Michalski et al., 1986) and ABACUS-2 (Greene,1988), and expert database systems (Kerschberg, 1986; Kerschberg, 1987; Kerschberg, 1988). The above systems have influenced the development of INLEN. INLEN also draws upon the experiences with AGASSISTANT, a shell for developing agricultural expert systems (Katz, Fermanian & Michalski, 1987), AURORA, a general-purpose PC-based expert system shell with learning and discovery capabilities, designed by Michalski and Katz [INIS, 1988], and EMERALD (Kaufman, Michalski & Schultz, 1989; Kaufman, Michalski & Schultz, 1990) a user-oriented multi-program environment for education and research in machine learning.

The main aims of this paper are to briefly present the architecture and the design of the INLEN system, explain its underlying principles, and to demonstrate some of its capabilities. Section 2 gives examples of some knowledge discovery tasks for which machine learning techniques seem to be particularily useful. Section 3 introduces INLEN's architecture, and shows how the system would addresses the problems brought up in Section 2. Section 4 focuses on the first experimental implementation of the system, called INLEN-1. Section 5 describes the results of applying INLEN-1 to searching for patterns in a database of scientific publications. Finally, Section 6 summarizes the main ideas of the paper and discusses issues for future research.

Examples of Knowledge Extraction from Data

This section gives examples of problems of knowledge extraction from data for which symbolic machine learning methods seem to be particularily useful. Data are assumed to be represented in a relational database table. In such a table, each row ("tuple") corresponds to some entity (object, event, example, etc.), and each column corresponds to an attritute used to characterize the entities. Each entry contains a value of the corresponding attribute as applied to given entity. If the value of some attribute is unknown for some entity, the corresponding entry is marked "?"; if some attribue does not apply to an entity, the corresponding entry is marked "N/A" (Non-applicable).

Figure 1 presents an example of such a relational table. The table characterizes different trains, using such attributes as the number of cars ("#Cars"), the type of load ("Load"), the type of engine ("Engine"), the number of engine wheels ("#EngWls"), the first name of the conductor ("Condctr"), the first name of the engineer ("Engineer"), the type of the car ("CarType"), the train line ("Line") and the train's direction ("Direction"). For example, train #1 is described as a passenger train with 7 cars, a diesel engine, etc. Train #3 does not have a conductor. Train #6 is a nuclear monorail, has no cars, and does not run on any regular train line.

It is assumed that the system has also encoded *background knowledge* about the train's attributes, specifically, about the values that each attribute can take, the type of each attribute (that characterizes the underlying order of values; e.g., linear, hierarchical, etc.), and about existing relationships among the attributes. An example of a relationship between attributes is: "If a train has no cars, then the attribute "the name of the conductor" does not apply, which is represented by a rule: ***Conductor** = N/A if **# Cars** = 0*. In general, there could be all kinds dependencies among attributes. Given such a table, the problem is to discover "interesting" relationships about the objects described in the table. Suppose, for example, that a data analyst wants to see is there is an interesting relationship between the train's direction and other train properties.

Here is a sample of general rules characterizing this relationship (found by an inductive learning program AQ15):

Direction is South, if (**Engine** = Electric and **# Cars** < 7) or (**Engine** = Coal and **CarType** = Flat)
Direction is North, if (**#EngWls** = 6 or 10)
Direction is West, if **# Cars** > 16
Direction is East, if (**#EngWls** = 8, or 12..16) and (**CarType**≠Flat) and (**Load** ≠ Lead or MilSupp or Food)
Direction is unknown, if (**Engine** = Nuclear) or (**Load** = MilSupp)

	Attributes								
Tuples	Train #	# Cars	Load	Engine	#Eng Wls	Condctr	Engineer	Car Type	Line	Direction
	1	7	Passngr	Diesel	10	Eric	Mike	Cl_box	Amtrak	North
	2	12	?	?	6	Ken	Casey	Tank	UP	North
	3	7	Wheat	Coal	8	N/A	Jerzy	Closed	B&O	East
	6	0	N/A	Nuclear	1	N/A	Alan	N/A	N/A	?
	7	4	Lead	Electric	12	Bart	Debbie	?	ATSF	South
	9	15	Fruit	Diesel	8	Paul	John	Cl_box	C&O	East
	10	37	Cars	Diesel	18	Joan	Steve	Flat	Penn	West
	11	33	CD's	Electric	4	Dale	JJ	J-top	Rio	West
	15	9	Passngr	Coal	8	Bill	Ashraf	Flat	ICG	South
	19	83	MilSupp	?	12	N/A	?	Tank	N/A	?
	20	6	Fuel	Coal	6	Janet	Lou	Coal	NCRR	North
	22	17	Tox_Wst	Coal	13	Tom	Jim	Op_box	?	West
	26	23	Animals	Coal	9	Barney	Richard	Cl_box	ICG	West
	41	8	Passngr	Electric	16	James	Michael	Cl_box	Metro	East
	42	9	Passngr	Electric	12	Stanley	Jerry	Op_box	Conrail	East
	44	15	Gold	Diesel	10	Janusz	Terry	Armored	L&N	North
	47	6	?	Electric	18	Gordon	Betsy	P-Top	B&M	South
	63	7	Clothing	?	16	Marvin	David	?	SP	East
	75	22	Food	Diesel	16	George	Carol	Closed	CB&Q	West
	84	15	Oil	Electric	14	Chuck	Victor	Tank	MP	East
	102	5	Animals	Coal	18	Bonzo	Bob	Flat	Conrail	South

Table 1. An exemplary relational table in INLEN describing a set of trains

The rule for the "Direction is South" includes, as one of the conditions, the engine's type. Suppose that engine's type is difficult to determine, and it would be better to have a rule that includes another attribute that is known. Here is an example of such a rule:

Direction is South if (**Eng Wls** $\geq$ 8 and **# Cars** $\leq$ 6) or (**# Cars** $\leq$ 9 and **CarType** = Flat)

The above examples illustrate problems of determining logical rules characterizing dependencies among any variables. Such problems are well handled by symbolic inductive learning programs, such as those in the AQ family (Michalski & Larson, 1978; Michalski & Larson, 1983).

Another class of problems is to create classifications of given entities. A traditional way of grouping (clustering) entities into classes is based on a measure of similarity among entities. Such an approach creates goupings, but does not provide any description (or justification) of them. Such a description has to be developed by a data analyst. The INLEN's approach is to implement a *conceptual clustering* method that groups entities on the basis of *conceptual cohesiveness* (Michalski, Stepp & Diday, 1981; Michalski & Stepp, 1983). Such a method produces groupins (clusters) and their descrtiptions. It can construct a number of alternative goupings. Below is an examle of a classification of the entities in the table developed by a conceptual clustering program (CLUSTER/2). The trains are grouped into four classes, Class 1, Class 2, Class 3 and Class 4, each characterized by the following self-explanatory description:

Class 1: Engine is Electric and CarType is Open
Class 2: Engine is Electric and CarType is Closed
Class 3: Engine is Coal or Diesel and CarType is Closed
Class 4: Engine is not Electric and CarType is not Closed

If a table is very large, the data analyst might want to reduce the number of rows (and/or columns) of the table by determining the most representative entities (and/or attributes) for any given class. In INELN, such tasks are performed by adopting some existing programs for this purpose (Cramm, 1983; Davis, 1981; Michalski & Larson, 1978). Also, the attributes initially given may not be sufficiently relevant to the given problem, and there is a need to generate new, more relevant attributes. This is the task of *constructive induction* (Michalski, 1983). Papers by (Wnek and Michalski, 1991a; Bloedorn and Michalski, 1992) describe recent powerful learning programs for constructive induction that are to be incorporated into INLEN.

Another type of data analysis problems is to seek quantitative rules (equations) that characterize the numerical attributes in the table. A more general problem is to formulate such equations and to determine their applicability conditions expressed in terms of qualitative attributes. Such a capability can be implemented by applying the machine discovery program ABACUS (Falkenhainer & Michalski, 1990; Greene, 1988).

As shown above, a data table may have some entries missing. An important problem is to estimate the most likely value for the unknown entry. The INLEN's approach is to use for this task an approach based on the theory of plausible inference (Collins & Michalski, 1989). In this approach, the likely value are estimated by conducting different "lines of reasoning" that employ plausible deduction, induction or analogy (Dontas, 1988). If the data are represent facts changing in time, there may be a need to analyze the time-based sequences in the data, and predict future values of the variables in the sequences. A "qualitative prediction" program such as SPARC (Michalski, Ko & Chen, 1985; 1986) applies to such a task.

In general, there is potentially a wide variety of data analysis tasks for which machine leanrning techniques can bring interesting new solutions. The research on INLEN's aims at integrating all kinds of such learning and knowledge discovery capabilities within a uniform, user-oriented global environment.

The Architecture of INLEN

As mentioned above, INLEN combines data base, knowledge base and machine learning capabilities in a single system. INLEN's design incorporates ideas from the recently developed expert database technology to combine the storage and access capabilities of a data base system with the ability to derive well-founded conclusions from a knowledge-based system (Kerschberg, 1986; 1987 and 1988). Among novel features of INLEN is the implementation of various advanced machine learning capabilities as easy-to-use operators. Until now, these capabilities existed only as separate experimental programs. By integrating such capabilites in one system, a user will has an access to a very powerful and versatile data analysis tool.

Figure 1 presents the top level architecture of INLEN. Main components of the system are a relational data base 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 data base and/or knowledge base. This knowledge base can contain not only knowledge about the contents of the data base, but also metaknowledge for the dynamic updating of the knowledge base itself.

The purpose for integrating the above capabilities is to provide a user with a set of advanced tools to search for useful knowledge from a data base, to organize that knowledge from different viewpoints, to test this knowledge on a set of facts, and to facilitate its integration within the original knowledge base.

These tools are designed to complement one another, and to be capable of performing many types of data analysis. For example, different operators might be employed to learn a set of rules from examples (empirical induction), generalize a descriptor or a set of objects (constructive deduction or induction), hypothesize explanations for events in the data based on rules in the knowledge base (abduction), speculate on unknown attribute values of an object based on known values of similar objects (analogical reasoning), and suggest unknown attribute values by employing rules or formulas in the knowledge base (deduction).

The data base consists of relational tables (RTs). The knowledge base consists of *knowledge segments* (KSs). A knowledge segment can be simple or compound. Simple KSs are in the form of rulesets, equations, networks and hierarchies. Compound KSs consist of combinations of any of the above, or combinations of KSs and RTs. In this way, a knowledge segement may represent, e.g., a set of rules from the knowledge base and a collection of facts from the data base (a set tables, attributes, their domains, etc.) to which these rule apply. For example, the conceptual clustering operator, given a relational table, creates a new relational table (stored in the data base) that represents clusters of entities, and a set of rules (stored in the knowledge base) that characterize these clusters. Another example of a KS structure is a relational table with a set of constraints and relationships among its attributes. Those 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 data base. 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.

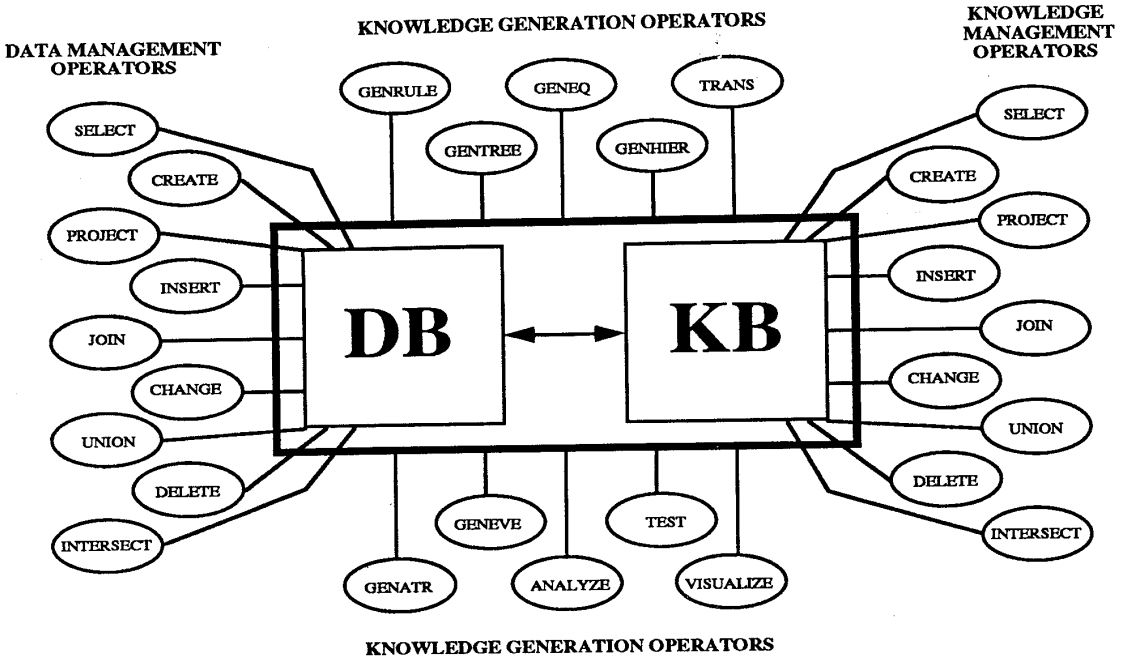

Figure 1. A top-level functional architecture of INLEN

INLEN employs four sets of operators: *data management operators* (DMOs), *knowledge management operators* (KMOs), *knowledge generation operators* (KGOs), and *macro-operators*. The data management operators are standard operators for accessing, retrieving and manually altering the information in the data base. Thus, they operate on RTs. The knowledge management operators 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 data base and the knowledge base. They take input from both the data base and the knowledge base, invoke various machine learning programs, and store results, as knowledge segements, in the data base and/or knowledge base. Figure 1 shows the ten major classes of KGOs. These classes and their individual members are discussed in the next section. The macro-operators are stored sequences of the other operators, along with instructions how to control the flow of commands within the macro. Macroperators allow a data analyst to develop complex data analysis programs that guide INLEN's knowledge discovery processes.

The schema shown in Figure 1 expands the initial design described in (Kaufman, Michalski & Kerschberg, 1989). The most important new features include several novel knowledge generation operators, the idea of macro-operators and conceptual data analysis programs. Also, the latest design groups the individual machine learning operators into higher-level operator classes based on their basic functions.

Knowledge Generation Operators (KGO)

As mentioned above, Data Management Operators (DMOs) and Knowledge Management Operators (KMO) are standard data and knowledge handling operators. Their description can be found in (Michalski, Kerschber, Kaufman and Ribeiro, 1992). Here we will concentrate on describing Knowledge Generation Operators (KGO) that are central to understanding the capabilites of INLEN.

The KGOs perform inferences on knowledge segments in order to create new or better knowledge. As part of their function, the KGOs also include implicit primitives to handle the retrieval of inputs and the placement of their results into the data and/or knowledge base. These structures are generally associated with compound KSs which include indexing tables within the knowledge base.

Each knowledge generation operator requires certain background knowledge and parameters. The background knowledge consists of information about the domain, the variables in the data tables, etc. The parameters define

criteria for choosing an output description among multiple candidates. For simplicity, these inputs are not included in the following descriptions, as they are assumed to be part of any KGO.

It is practical to group the KGOs into classes by the type of output they generate (Figure 1.) Those whose primary output consists of rules are separated from those that only generate tuples. Operators in each class are discussed below. Most of these operators are extensions of existing programs.

GENRULE: Generate Rules

Operators in the GENRULE class take some data and/or knowledge as an input, and return a ruleset consisting of facts induced from the input examples. The generated rules consist of a decision part implied by a condition part. The decision part consists of a conjunction of one or more statements or actions, while the condition part consists of a disjunction of conjunctions, each consisting of one or more elementary conditions. Specific GENRULE operators differ from one another in the organization of the input (unordered or ordered) and/or the type of rules generated (characteristic or discriminant). The GENRULE KGOs include the following operators:

CHARSET (Characterize Set) determines a description characterizing a class of entities. Input to the operator may be a table representing a group of events and their relevant attributes. It may also be a set of knowledge segments, defined by their own meta-attributes, that the user wishes to characterize with a rule. CHARSET discovers characteristic rules that describe all of the examples in the input group in as much detail as possible. The output from this operator includes the input set of events in addition to the generated rules that describe the characterization.

DIFFSET (Differentiate Set) takes one set of objects (each object represented as a tuple in a relational table that may represent data or metaknowledge) as a primary input, and one or more sets of objects as a controlling input. These sets may be represented by separate RTs, or by the values of one or more "decision variables" within a single table. DIFFSET induces general rules that encapsulate the differences between the primary set and the other classes. The operator may be called upon to treat each of the groups in turn as the primary and discover rules differentiating it from the others. The output KS consists of the ruleset created by the operator, and the object classes, represented by RTs. Here, the emphasis is on finding discriminant descriptions, i.e., descriptions that specify sufficient conditions for distinguishing one class of objects from the other class(es). This operator is demonstrated in Section 3.

CHARSET is based on research on learning characteristic descriptions from examples. An early program for this purpose, UNICLASS, is described in (Steep, 1979). A version of the program that executes both CHARSET and DIFFSET, AQ11, is described in (Michalski & Larson, 1983). It includes capability for a "no-memory" incremental learning. The next major program in this family is AQ15, which includes a capability for "full-memory" incremental learning and the TRUNC method for two-tiered concept representation (Hong, Mozetic & Michalski, 1986). More recent research has resulted in the incorporation of further capabilities into the AQ system, such as learning flexible concepts (Bergadano et al., 1992; Zhang, 1992) and capabilities for constructive induction (Bloedorn & Michalski, 1992; Wnek & Michalski, 1991a).

CHARSEQ (Characterize Sequence) determines descriptions characterizing a sequence of objects or events. This is a more complex operator than CHARSET, since the learner must now take into account the influences of ordering and positioning of examples in the sequence, and it may also have to consider negative examples -- objects that do not belong at a given point in the sequence. The input consists of a table containing the examples in the sequence, including an example's location in the sequence. The output from CHARSEQ consists of a ruleset characterizing the sequence.

DIFFSEQ (Differentiate Sequence) discovers differences between two or more sequences of objects or events. Input consists of a primary sequence and one or more other sequences. The operator will seek rules that encapsulate the differences between the primary sequence and the other input sequences. Like CHARSEQ, DIFFSEQ is far more complex than its other counterpart, and it returns a ruleset describing its discoveries.

CHARSEQ and DIFFSEQ represent an extension of the SPARC methodology for determining patterns in sequences, described in Diettrich & Michalski, 1986; Michalski, Ko & Chen, 1985; 1986). The methodology assumes that individual entities in the sequence are described by a finite set of multivalued and multitype attributes. SPARC employs three rule models for representing and discovering patterns in various types of sequences: a decomposition model that captures direct dependencies of the future event on the past events, a periodic model that expresses a periodic behavior of a sequence, and a DNF "catch-all" model. Due to the enormous complexity of the prediction problem, the implementation of these operators requires a substantial amount of new research. Ongoing research involves enhancing and extending the SPARC's rule models, and their recursive evocation. These two enhancements will enable the system to encompass a much wider classes of sequences, and to improve its prediction capabilities.

GENTREE: Generate Decision Trees

The two GENTREE operators output knowledge in the form of decision trees. EVENTREE (Event to Tree) uses events in a relational table as input, and generates a tree for classifying the input events. RULETREE (Rule to Tree) organizes a set of decision rules into one or more trees. EVENTREE is based on ideas implemented in the C4.5 program for generating decision trees from examples (Quinlan, 1990) and ASSISTANT (Cestnik, Kokonenko & Bratko, 1987). The RULETREE operator utilizes the OPTTREE program for creating decision trees from rules (Layman, 1979; Michalski, 1978).

GENEQ: Generate Equations

GENEQ is a single operator that discovers equations that describe numeric data in a set of examples, and formulates conditions for applying these equations. The input to GENEQ includes a table that incorporates quantitative data and constraints on the mathematical operations that may be used to manipulate these values. GENEQ returns a set of equations, and a set of rules that characterize the elements of the input table to which they apply.

GENEQ is based on the ABACUS-2 system for integrated qualitative and quantitative discovery (Greene, 1988), an extension of ABACUS (Falkenhainer & Michalski, 1990). These quantitative discovery programs are conceptually related to systems such as COPER (Kokar, 1986) BACON (Langley, Bradshaw & Simon, 1983) and FAHRENHEIT (Zytkow, 1987).

GENHIER: Generate Hierarchies

The GENHIER operators organize a set of tuples, rules, equations, etc. into a set of clusters or a hierarchy. The CLUSTER operator creates a logical division of the input objects into two or more groups (a hierarchy one level deep). The TAXONOMY operator generates a full-fledged classification hierarchy by a recursive invocation of CLUSTER. In addition to the generated hierarchies, both operators determine a set of rules characterizing the created groups. The rule characterizing the top-level group (the set of all input examples) is equivalent to the rule that would be generated by applying CHARSET to the input set. Rules on lower levels emphasize the differences between these rules and their parents and siblings in the classification hierarchy.

The internal form of the output from these operators is a knowledge segment consisting of a relational table and a ruleset. The table is an extension of the input table, with additional columns specifying the classes the examples have been put into on each level of the hierarchy. The rules characterize the individual groups and the input RT as a whole. CLUSTER and TAXONOMY use the conceptual clustering algorithm implemented in the CLUSTER program described in Michalski, Stepp & Diday, 1981; Michalski & Stepp, 1983; Stepp, 1983; Stepp, 1984.

TRANS: Transform Knowledge Segments

The TRANS operators perform basic inferential transformations on knowledge segments, hence both the primary inputs and outputs are knowledge segments of the same type, typically decision rules. There are two pairs of inverse operators: ABSTRACT and CONCRETIZE, and GENERALIZE and SPECIALIZE, in addition to IMPROVE, an operator that improves knowledge by exploring additional examples or facts. Default, induced or user-specified parameters guide the system in selecting from multiple possible outputs.

ABSTRACT modifies its input knowledge segment by removing details from its description. For example, the known fact, "The Chrysler Dynasty is a mid-sized car that gets 26 miles per gallon", may be abstracted by replacing concepts in it by more general concepts, entailed by the original one. The resulting knowledge segment might contain the fact, "The Chrysler Dynasty is an efficient automobile for its class." Conversely, CONCRETIZE takes a fact such as "The Lincoln Town Car is a luxury automobile" as input, and creates an output statement such as "The Lincoln Town Car is expensive, and contains many comforts and conveniences for the driver and passengers."

GENERALIZE and SPECIALIZE affect the size of the set covered by an input KS. For example, the rule, "A creature is in class 1 if it is a dog," can be generalized to "A creature is in class 1 if it is a mammal," or specialized to "a creature is in class 1 if it is a golden retriever."

The methods for implementing ABSTRACT and CONCRETIZE operators have not been much studied in the past. A discussion of their function and examples are in (Michalski, 1990). The input to IMPROVE is one or more knowledge segments and a new set of examples. From the examples, any exceptions to the input knowledge are detected, and the KSs are modified accordingly by a learning program. The output from this operator consists of the revised rules. The methodology for IMPROVE is based on the AQ15 program (Hong, Mozetic & Michalski, 1986). AQ15 has been implemented and tested in conjunction with other environments.

GENATR: Generate Attributes

The GENATR operators map relational tables to relational tables whose rows are the same but whose columns have been changed, either by the addition of new attributes or by the removal of old ones.

SELATR (Select Attributes) determines the attributes in a relational table that are most relevant for differentiating between various classes of objects, and produces a reduced table that retains only variables chosen by the operator (Figure 2). By keeping only the most relevant attributes in the object (example) descriptions, one can significantly reduce the computation time required by operators such as CLUSTER or DIFFSET.

CONATR (Construct Attribute) applies operators specified by its arguments to combine given attributes into useful composites. The output consists of an expanded table that includes the new composite variables, and equations specifying the relationship between the created variables and the variables from which they were derived (Figure 2). For example, CONATR may create the sum or product of two numerical attributes, and assign to them attribute names, so they can be used as new attributes.

The GENATR operators are based on existing programs. SELATR employs the VARSEL program (Baim, 1982), and CONATR incorporates the capabilities of the CONVART program (Davis,1981).

GENEVE: Generate Events

The GENEVE class covers a wide variety of operators that generate a new set of tuples, either from an existing relational table, or from the entire event space to which a table's tuples belong. The events in the output table are selected according to some criterion of desirability, such as typicality, extremity, being contained in two or more classes, etc.

SELEVE (Select Event) determines the examples (objects) that are the most representative of the examples contained in input relational tables. The output from this operator is a subtable of the input table, with the same columns as the original table. The most promising examples are returned in this reduced relational table, while other examples are rejected (Figure 2).

CONEVE (Construct Event) searches the example set or event space for elements satisfying an input description, and satisfying some selection criteria such as those described above. In many cases, CONEVE will chain to another operator whose specialty involves the characteristics being sought. For example, in projecting next month's sales based on previous figures, CONEVE might call upon CHARSEQ to infer patterns in the sequence of sales amounts. The input description will be treated as a knowledge segment, and the output will be one or more tuples, representing actual or hypothetical examples in a data set.

PREDVAL (Predict Value) speculates on likely values for unknown attributes of incomplete or hypothetical data elements. It comes to its conclusions using existing knowledge segments to reason about the incomplete tuples and plausible reasoning to learn from examples of similarly behaving events, and sequence characterization techniques to extrapolate from changes in a linear domain. Input consists of incomplete tuples or, in the case of hypothetical data, a characterization of some of the would-be tuple's attributes. The operator's output is one or more tuples.

SIMILIZE (Find Similar) seeks out events or relationships that are similar to the input in some defined sense. A similar example may have similar attribute values or relationships among its attributes. A similar concept may involve similar attributes or ranges of values. Input to SIMILIZE may be a tuple, a table, or a knowledge segment, and the output will generally be one or more structures of the same type as the input. Depending on the application, the output can be generated from existing structures in the data or knowledge base, or from the entire event space.

SELEVE uses the ESEL methodology, described in (Cramm, 1983) (Michalski & Larson, 1978). PREDVAL, CONEVE, and SIMILIZE are the subject of research, which is based on the foundations set up by the SPARC and APPLAUSE (Dontas, 1988) programs.

ANALYZE: Analyze Data

The ANALYZE family of operators return knowledge in the form of numerical weights characterizing various logical or statistical relationships.

RELATR (Relate Attributes) determines a relationship, such as equivalence, implication, correlation or monotonic dependency that may exist between two or more attributes in a relational table. Input to RELATR consists of the tables and attribute specifications, and may also include a specification of context, such as "Only compare attributes for monotonicity over events in which another attribute is constant." The output from this operator is a knowledge segment that links the inputs with any discovered relationships, and includes a numerical strength of the relationships.

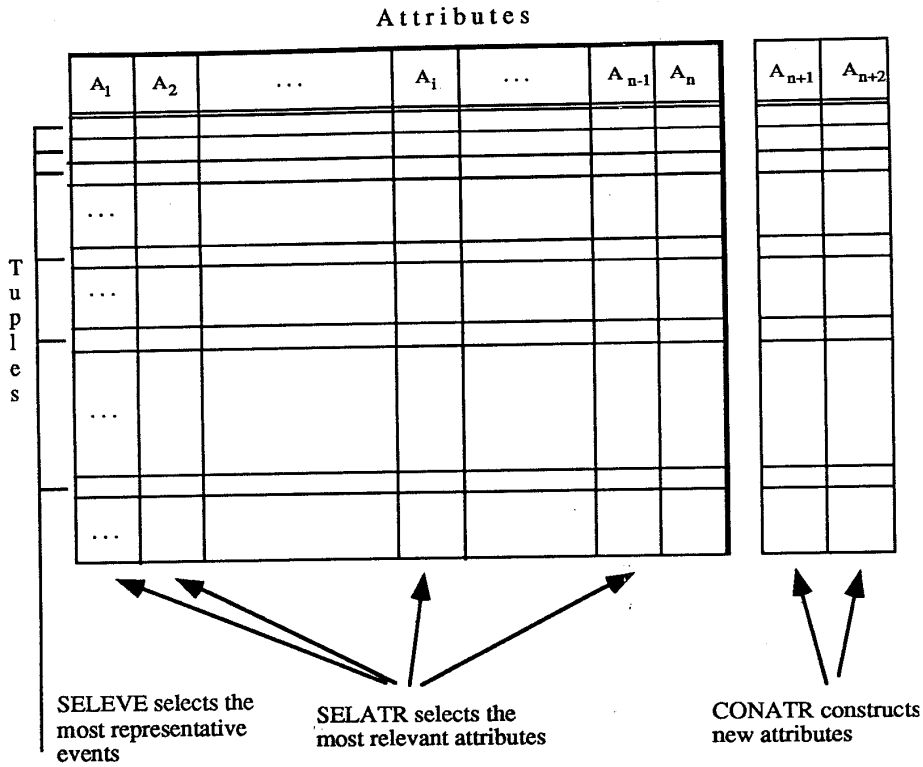

Figure 2. An Illustration of the roles of the SELEVE, SELATR and CONATR operators

RELEVE (Relate Events) behaves similarly to RELATR, but it determines relationships among tuples in a relational table. The input to RELEVE consists of a table of the input events, and the context in which the examples are to be compared. The output consists of a knowledge segment that includes links between examples and weights that represent the strength of the relationships.

RELKS (Relate Knowledge Segments) discovers relationships such as inclusion, disjointedness, correlation, generalization and abstraction within a set of knowledge segments. The input to RELKS consists of a set of knowledge segments and a description of the context in which relationships are to be discovered. The operator returns a KS consisting of a table of its results and a description of the discovered relationships.

GENSTAT (Generate Statistics) performs a statistical analysis and computes various statistical properties of the elements in a relational table. This is actually a meta-operator that represents various specific statistical operations, such as finding means, standard deviations, correlation coefficients, average rates of change, etc. Input consists of a relational table, while output consists of a KS that contains a table of results and a link between that table and the input table.

TEST: Test Knowledge

The TEST operator determines the performance of a ruleset on a set of examples by testing the input knowledge segments for consistency and completeness with regard to the input examples (specified in a relational table). Consistency implies that no event in the example set is covered by rules of two or more different classes. Completeness refers to the condition that every example is covered by the conditions applying to at least one rule. Input consists of a set of examples to be tested (in the form of an RT) and a set of knowledge segments that are to be tested against the examples. The output KS consists of several relational tables containing TEST's analysis, including parameters characterizing the performance of the tested knowledge segments. The primary output table is in the form of a *confusion matrix*, i.e. a matrix whose $(i,j)^{th}$ element shows the percentage of the testing examples from the class i that were classified by the rules to be in class j. TEST also creates links between the output tables

and the input structures. TEST uses the ATEST methodology (Reinke, 1984) for analyzing consistency and completeness in rules, and generating confusion matrices.

VISUALIZE

VISUALIZE presents a graphical representation of the data or rules. It employes two operators. One is DIAV (which stands for **dia**grammatic visualization), and the second is GRAPH (which stands for **graph**ical visualization). The DIAV operator is for visualizing tuples, rules, and operations on them using a diagram, which is a plane representation of a mutlidimentional space spanned over discrete attributes. The GRAPH operator is based on traditional techniques for visualizing numerical data and continuous functions.

The DIAV operator takes as input a relational table, or a knowledge segment representing a rule. The output appears as a two-dimensional representation of the event space, with the input set highlighted. Figure 3 shows a diagrammatic representation of two rules (A and B) constituting a description of Class 1. The class consists of examples of robot images characterized in terms of four attributes: HeadShape, JacketColor, Holding, and IsSmiling. The attribute "HeadShape" specifies the shape of the robot's head, which can be **R**ound, **S**quare or **O**ctagonal. The attribute "JackeColor" takes four values: **R**ed, **Y**ellow, **G**reen or **B**lue. The attribute "Holding" specifies what the robots is holding in its hand, which can be **S**word, **B**alloon or **F**lag. The attribute "IsSmiling" can be **T**rue or**F**alse. Each cell in the diagram represents a unique combination of atrribute values. The VISUALIZE-DIAV operator is based on the DIAV program (Wnek & Michalski, 1991b; Wnek et al, 1990).

MACRO-OPERATORS

Macro-operators allow for repeatable, standard sequences of operations. They encompass a small number of INLEN operators, and can be added to a KGO menu and called upon as single operators. As with the basic INLEN operators, macro-operators can be invoked in conjunction with any appropriate parameters, arguments or specifications. For example, a macrooperator might call for the automatic generation of a statistical and similarity analysis, and a comparison with predicted levels, upon the receipt of a company's sales data for a new month.

It may also be the case that there is an application, possibly repeatable, that must call upon a longer sequence of operators, possibly making simple control decisions based on the output of earlier operators in the sequence. INLEN allows the user to read a data-analysis program from a file in order to perform such tasks. Because such programs of operators can make their own control decisions, they allow for long, unsupervised sessions. The language for these programs includes the capacity for branching, looping, and local variables. For example, a program may be called to invoke a DMO for adding new records to a data base from a file, until all records in a waiting area were cleared out. It can then call TEST to see if the new records are consistent with the relevant knowledge stored in the knowledge base. In the case of inconsistency, it can then call the DIFFSET operator to modify the inconsistent knowledge.

Implementation of INLEN

The design and implementation of INLEN builds upon our earlier development of QUIN (Michalski & Baskin, 1983; Michalski, Baskin & Spackman, 1982; Spackman, 1983), ADVISE (Baskin & Michalski, 1989), and AURORA (INIS, 1988) systems. As described above, the complete INLEN system will include a great variety of modules, some of which can be used as powerful stand-alone programs. In order to implement such a large-scale system with so many capabilities, we have undertaken its development in stages. Each stage represents a version of the system with an increasing set of capabilities. Each version is self-contained, and can be independently tested and applied to data analysis problems.

The first stage, already completed, is the implementation and testing of INLEN-1. Below is a brief description of this version. INLEN-1 includes a knowledge base of decision rules, a relational data base, and an extensive user-oriented menu-based graphical interface. INLEN-1 has six major modules:

- Definition of an application system
- Knowledge acquisition
- Rule learning and discovery
- Advisory and prediction
- Review of an application system
- Tutorial on using INLEN-1

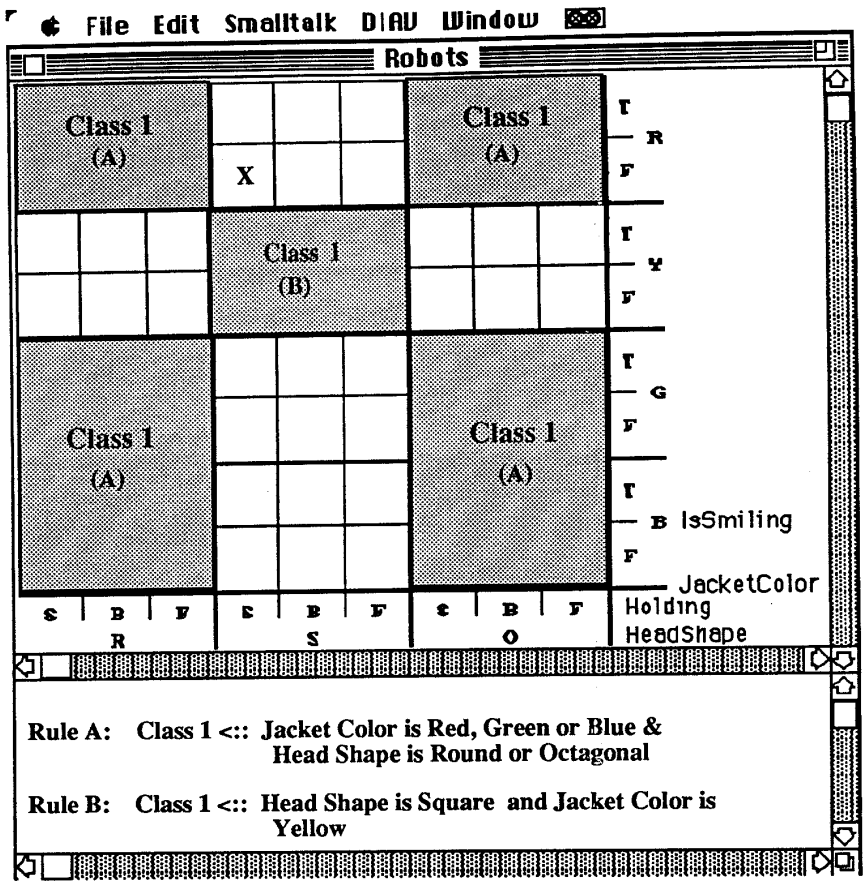

Figure 3. An example of the diagrammatic visualization of rules

The application system definition module allows user to define the data base schema, import facts into the data base from other data bases, and specify various parameters to control learning and advisory modules. Attributes in the data base may be defined as *nominal*, *linear* or *hierarchical*. Domains of nominal attributes are sets of values whose order is arbitrary, and thus does not carry information. For example, the "automobile license number" is a nominal attribute. Domains of linear attributes represent totally ordered sets. For example, any numerical measurement, such as length, height, temperature, is a linear variable. Domains of hierarchical attributes are hierarchies. For example, if malfunctions of a complex system are organized into a hierarchy of different types, then "malfunction type" is a hierarchical attribute. Knowing the type of an attribute helps the system to appropriately conduct reasoning involving this attribute, for example, in inductive generalization operations and in matching instances with rules.

The knowledge acquisition module consists of programs that support direct entry of rules and other knowledge into the system, as well as their modification and improvement. The module includes a rule editor that helps to fomulate rules accoriding to a certain format. The learning and discovery module contains programs supporting various knowledge generation operators (KGOs). The KGOs currently integrated in the system include: CHARSET, DIFFSET, IMPROVE, TEST and PREDVAL. These operators support rule learning from examples, rule optimization, rule improvement through examples, and rule testing for completeness and consistency.

The advisory and prediction module conducts reasoning utilizing the inputs provided or requested from a user, and the knowledge base. The goal of reasoning is to determine advice regarding a decision, e.g., a classification of an unknown entity, a diagnostic decision, a choice among alternative solutions, etc. The process can be viewed as a

determination of an unknown value of a *output* attribute. Any attribute in the data can be the output attribute. The choice is made by the user.

The inference process takes into consideration various uncertainties, so that any generated advice is associated with degree of confidence in it. The system usually generates a list of alternative solutions with associated degrees of confidence. The solutions are developed through a three-stage inference process. The first, *reduction* stage, reduces the set of possible hypotheses by testing them against facts voluntarily supplied by a user.

The second, *discrimination* stage, generates consecutive questions to the user, collects answers and propagates them through the system. The questions asked are determined on the basis of three criteria: knowing an answer to the question should maximally reduce the number of candidate hypotheses, the question should be an "important" one as determined by a domain expert, and the question should be "logically" connected to other questions.

The third, *confirmation* stage, takes a leading candidate hypothesis, and attempts to either confirm it or disconfirm it by collecting more information. The confirmation is achieved if the candidate's confidence level either exceeds a user-defined threshold, or exceeds the confidence degree in all other candidate hypotheses by a predefined amount. The system review module consists of utilities to allow the user to view the contents of the data base and the knowledge base, the data base schema, and the learning and inference parameters.

INLEN 1 was tested on a variety of problems requiring rule discovery from facts or rule improvement through facts. One experimental application is described below.

Experimental Application

This application concerns the analysis and search for patterns in a data base of scientific publications written by scientists in the Commonwealth of Independent States (CIS; formerly USSR). The data base, which we refer to as CISA (CIS Authors), contains 2,841 records. Each record contains information on a particular paper published by a CIS author. Some records contain values for all fields, while other records have missing or non applicable values.

Data Base Definition

Currently, INLEN-1 only supports analysis of one relational table at a time. However, this table may be formed from data contained within multiple tables. In the CISA table created for the experiments, every scientific publication is described by a record listing the values of the following attributes:

1. AUTHOR - Author's name.
2. COAUTHOR - Co-authors' name.
3. TITLE - Title of the paper.
4. PUBYR - Publication year of the paper.
5. INSTITUTE - Research institute, or other affiliation of the author.
6. SPECIALTY1 - Principal specialty topic of the paper (e.g. COMMS, RADAR, etc.).
7. SPECIALTY2 - Secondary specialty topic of the paper.
8. SPECIALTY3 - Tertiary specialty topic of the paper.
9. SYSTEM - Physical system mentioned in the paper (e.g. Salyut, Halley Probe).
10. AGENCY - Collecting agency within the U.S.

Here is an example of a record instance in the CISA data base:

1. AUTHOR = Aleksandrov, Y. M.
2. COAUTHOR = Kotelnikov, V. A.; Andreyev, R. A.; Zaytsev, A. L.
3. TITLE = Results of Radar Observation of Venus on Wave Length of 39 Centimeters in 1980
4. PUBYR = 1980
5. INSTITUTE = Institute of Radio Engineering and Electronics
6. SPECIALTY1 = Radar
7. SPECIALTY2 = Propagate
8. SPECIALTY3 = N/A
9. SYSTEM = Venus Mapper
10. AGENCY = DTIC.

Experiments and Results

One goal of this experimental application was to determine general rules and "interesting" patterns and general rules characterizing the CISA data base. The second goal was to use the acquired patterns or rules to fill in any missing

data in the INSTITUTE field. The rules needed for the latter goal are expressed as relationships between the INSTITUTE field and the other attributes in the CISA data base.

Our approach toward this goal was to first select only those records with a known institute since the author's research affiliation was not always listed on the paper and resulted in missing values for this field. This led to a view of the CISA data base with 179 records. Using this view, we then generated rules for each institute in our data base. This was accomplished by applying INLEN's inductive capabilities against all records to determine relationships about the CIS institutes. These relationships were then stored in INLEN's knowledge base where they were used by INLEN to deduce the missing values for the INSTITUTE field.

To illustrate how INLEN-1 represents knolwedge, below is an exmple of a *characteristic* rule learned by it for the CISA domain:

	#Pos	#Neg	Dis	Com
Institute is Leningrad Institute of Materials and Optics if:				
1. SYSTEM is Halley Probe	3	0	100	100
2. SPECIALTY1 is Optics	3	1	75	100
3. AGENCY is JPRS or SNAP	3	7	30	100
4. PUB_YR is 1984	3	29	9	100

A characteristic rule lists all conditions characterizing a set of examples, in contrast to discrimiant to rule that lists only those sufficient for discrimination (Michalski, 1983). The weights associated with the conditions in this rule are generated automatically during rule learning, based on the example set from which the rule was learned. The #Pos and #Neg weights represent the total number of positive and negative examples of the class that matched the condition. For example, three examples of the LITMO class had Optics as their primary specialty, while one example of other classes also specialized in Optics. The Dis paremter ("distinctiveness") is equal to (100 * #Pos) / (#Pos + #Neg). It represents the percentage of examples having the given value(s) that belong to the positive class. Distinctiveness can be thought of a measure of how much information is provided by a given condition. The high distinctiveness indicates that if the primary specialty of an institute is Optics then this is strong evidence that this institute is LITMO. The low distinctiveness associated with PUB_YR suggests that knowing that a publication is from 1984 only provides flimsy evidence toward this conclusion.

The parameter Com ("commonality") represents the degree to which a condition describes the examples of a class. It is equal to (100 * #Pos) / (#Examples of the class), and may be less than 100, if several rules are required to cover the examples of a class. A condition with high commonality will be universally true throughout the examples of a given class, while one with low commonality will apply only to a small subset of the class. A goal when trying to ascertain missing values for given attributes is to gather information associated with high distinctiveness and commonality values; the former because it will discriminate better between classes, and the latter because it will be associated with more potential examples of the target class.

The analysis using INLEN led to several interesting discoveries about the CISA data base. One discovery was a rule which related the Halley Probe system with the Leningrad Institute of Materials and Optics. The actual rule produced by INLEN in this case was:

INSTITUTE is Leningrad Institute of Materials and Optics if: SYSTEM is Halley Probe.

This rule was generated by setting INLEN's knowledge creation parameters to generate the least complex rule possible. This setting was chosen since we were searching for simple but possibly illuminating descriptions of the CIS institutes.

Another interesting rule discovered by the system concerned the technical specialty areas of the Kaunas Polytechnical Institute. This rule was created with the same knowledge generation operator. The rule produced by INLEN-1 was:

INSTITUTE is Kaunas Polytechnical Institute if: SPECIALTY2 is Wideband or Satellites.

This rule illustrates the presence of disjuncts to form a simple rule, and tells us that research on wideband satellites occurs at this institute. These results illustrate just two of the interesting relationships discovered from the CISA data base using INLEN-1.

Once relationships were found for all the institutes and our knowledge base for INSTITUTE was complete, we then attempted to deduce values for those records with missing institute values. We were not always able to infer these

missing institute values with a high degree of confidence, but in several cases, the discovered knowledge was sufficient to infer values for these institutes with very high degrees of confidence.

The experiments with the CISA data base provided us with an opportunity to evaluate some of the current capabilities of INLEN-1, and helped to suggest further topics for research in the context of the applications illustrated by the CISA experiments. Here is a brief summary of general findings:

- The system allows users with little domain expertise to discover interesting relationships in the data.
- It can discover simple or complex patterns that have been hidden by the volume of the data.
- It may find surprising relationships between attributes not known to be directly linked.
- The knowledge representation used in the system is easy to interpret and understand, and thus the knowledge discovered can be easily explained and related to other knowledge.

The experiments also exposed some problems of the current system that require new research, and an implementation of additional features. Currently, many of the manipulations required are manually applied to the data base. The system needs additional mechanisms that would help to automate some of the inference processes. Future experiments should address the problem of "scaling up" the system, that is to test its applicability to analyzing very large data bases. Finally, the system should have an ability for supporting an evolution of the the data base and knowledge base schema to be able to adquately incorporate new data and learned knowledge.

Conclusions and Future Research

INLEN is a large-scale multistrategy data-analysis system capable of performing a wide variety of inferential operations on data and knowledge. The system is designed to serve as an intelligent assistant for data analysis and discovery of interesting regularities in them. These regularities can be detected in qualitative data, quantitative data, in a mixture of qualitative and quantative data, or in the knowledge base itself. INLEN also provides functions that facilitate manipulation of both the data and the knowledge base.

The research on INLEN is intended to demonstrate the feasibility of a system capable of examining large quantities of data, detecting trends, complex correlations and anomalies, analyzing the importance of these discoveries, visualizing and reporting significant patterns, and predicting missing or future data elements. In business domains, a strategic advantage may be attained. In scientific domains, hidden regularities may be discovered. The ability to process and analyze increasing volumes of data can become a tremendous asset to users faced with more information than they can absorb. Using machine learning and inference techniques, the search through the data can be made in far less time, and with a greater "signal-to-noise ratio."

INLEN implements a number of novel ideas. It integrates a variety of knowledge generation operators that permit a user to search for various kinds of relationships and regularities in the data. This integration allows it to exploit the strengths of diverse learning and discovery programs, and to reduce the limitation to specific tasks. To achieve this integration, the concept of a *knowledge segment* has been introduced. The knowledge segment stands for a variety of knowledge representations such as rules, networks, equations, etc., each possibly associated with a relational table in the data base (as in the case of a set of constraints), or for any combination of such basic knowledge segments. INLEN also utilizes macro-operators and data analysis programs to facilitate operation of the system and to allow more flow control to be handled by INLEN itself. Users can easily develop and invoke both of these tool sets.

By employing diverse knowledge generation operators, INLEN has the capability for multistrategy learning and discovery. Depending on the situation at hand, operators may be called upon to perform empirical induction, constructive induction or deduction, abductive hypothesizing, analogical reasoning, or deductive inference. This research aims to create a domain-independent learning and discovery system that is not limited to a narrow scope of tasks, but is capable of assisting data base analysts in diverse fields.

Many of the INLEN operators are based on the research results and programs developed over the last fifteen years at our and other laboratories. Incorporating these programs into INLEN requires different amounts of effort. In a few cases, this includes primarily a change of the program interface. In other cases, the programs have to undergo major modifications or be redeveloped from scratch. Finally, some other operators are still at the stage of research and initial implementation.

In conclusion, the experiments conducted with the first version of the system, INLEN-1, have clearly demonstrated that it is a very flexible and powerful intelligent assistant for knowledge discovery in data.

References

Baim, P.W., "The PROMISE Method for Selecting Most Relevant Attributes for Inductive Learning Systems," Report No. UIUCDCS-F-82-898, Department of Computer Science, University of Illinois, Urbana IL, Sept. 1982.

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

Bergadano, F., Matwin, S., Michalski R. S. and Zhang, J., "Learning Two-tiered Descriptions of Flexible Concepts: The POSEIDON System," *Machine Learning Journal*, 8, pp. 5-43, 1992.

Bloedorn, E. and Michalski, R.S., "Data-Driven Constructive Induction in AQ17-DCI: A Method and Experiments," to appear in *Reports of Machine Learning and Inference Laboratory*, Center for Artificial Intelligence, George Mason University, Fairfax, VA, 1992.

Boose, J.H., and Gaines, B.R. (Eds), Proceedings of the 4th Knowledge Acquisition for Knowledge-based Systems Workshop, Banff, Canada, October, 1989.

Boose, J.H., Gaines, B.R., and Ganascia, J.G. (Eds), *Proceedings of the Third European Workshop on Knowledge Acquisition for Knowledge-based Systems*, Paris, 1989.

Boose, J.H., Gaines, B.R., and Linster, M. (Eds), *Proceedings of the Second European Workshop on Knowledge Acquisition for Knowledge-based Systems*, Bonn, June, 1988.

Boulanger, A. "The Expert System Plant/CD: A Case Study in Applying the General Purpose Inference System Advise to Predicting Black Cutworm Damage in Corn," M.S. Thesis, UIUCDCS-R-83-1134, Department of Computer Science, University of Illinois, Urbana, July 1983.

Bratko, I., Mozetic, I., and Lavrac, N., "Automatic Synthesis and Compression of Cardiological Knowledge," J. E. Hayes, D. Michie, J. Richards (Eds.) *Machine Intelligence II*, Oxford: Clarendon Press, pp. 435-454, 1988.

Cestnik, B. Kokonenko, I. and Bratko, I., "ASSISTANT 86: A Knowledge Elicitation Tool for Sophisticated Users," in *Proceedings of the Second European Working Session on Learning*, Bratko, I. and Lavrac, N. (Eds.), Bled, Yugoslavia, 1987.

Chilausky, R., Jacobsen and R. S. Michalski, "An Application of Variable-Valued Logic to Inductive Learning of Plant Disease Diagnostic Rules," Proceedings of the 1976 International Symposium on Multiple-Valued Logic, Utah State University, Logan, Utah, May 25-28, 1976.

Collins, A. and Michalski, R.S., The Logic of Plausible Reasoning: A Core Theory," *Cognitive Science*, Vol. 13, No. 1, pp. 1-49, 1989.

Cramm, S.A., ESEL/2: "A Program for Selecting the Most Representative Training Events for Inductive Learning," Report No. UIUCDCS-F-83-901, Department of Computer Science, University of Illinois, Urbana, IL, Jan. 1983.

Davis, J. H., "CONVART: A Program for Constructive Induction on Time Dependent Data," Master's Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1981.

Dietterich, T. and Michalski, R.S., "Learning to Predict Sequences," in *Machine Learning: An Artificial Intelligence Approach Vol. II*, Michalski, R.S. Carbonell, J.G. and Mitchell, T. (Eds.), Morgan Kaufmann Publishers, Los Altos, CA, 1986, pp. 63-106.

Dontas, K., "Applause: An Implementation of the Collins-Michalski Theory of Plausible Reasoning," Master's Thesis, University of Tennessee, Knoxville, TN, August 1988.

Falkenhainer, B. and Michalski, R.S., "Integrating Quantitative and Qualitative Discovery in the ABACUS System," in *Machine Learning: An Artificial Intelligence Approach, Volume III*, Kodratoff and Michalski (Eds.), Morgan Kaufmann, San Mateo, CA, 1990.

Fermanian, T. and Michalski, R.S., AGASSISTANT: A New Generation Tool for Developing Agricultural Advisory Systems," chapter in *Expert Systems In Developing Countries: Practice and Promise*, pp. 43 - 69, Charles K. Mann & Stephen R. Ruth (eds.), Special Studies in Social, Political & Economic Development, Westview Press Boulder & San Francisco & Oxford, Westviews 1992.

Fermanian,T.W., Michalski R.S. and Katz, B., "An Expert System to Assist Turfgrass Managers in Weed Identification," presented at the *Proceedings of the 1985 Summer Computer Simulation Conference*, Chicago, IL, July, 1985.

Greene, G., "Quantitative Discovery: Using Dependencies to Discover Non-Linear Terms," Master's Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1988.

Hong, J., Mozetic, I. and Michalski, R.S., "AQ15: Incremental Learning of Attribute-Based Descriptions from Examples, the Method and User's Guide," Report No. UIUCDCS-F-86-949, Department of Computer Science, University of Illinois, Urbana IL, 1986.

INIS (International Intelligent Systems, Inc.) "User's Guide to AURORA 2.0: A Personal Inference and Discovery System for Automated Rule Acquisition," Fairfax, VA, 1988.

Katz, B., Fermanian, T.W. and Michalski, R.S., "AgAssistant: An Experimental Expert System Builder for Agricultural Applications," Report No. UIUCDCS-F-87-978, Department of Computer Science, University of Illinois, Urbana, IL, October 1987.

Kaufman, K., Michalski, R.S. and Kerschberg, L., "Mining For Knowledge in Data: Goals and General Description of the INLEN System," IJCAI-89 Workshop on Knowledge Discovery in Data bases, Detroit, MI, 1989.

Kaufman, K., Michalski, R.S., and Schultz, A., "EMERALD-1: An Integrated System of Machine Learning and Discovery Programs for Education and Research, Programmer's Guide for the Sun Workstation," *Reports of Machine Learning and Inference Laboratory*, MLI 90-13, Center for Artificial Intelligence, George Mason University, Fairfax, VA, 1990.

Kaufman, K., Michalski, R.S., and Schultz, A., "EMERALD-1: An Integrated System of Machine Learning and Discovery Programs for Education and Research, User's Guide," *Reports of Machine Learning and Inference Laboratory*, MLI 89-12, Center for Artificial Intelligence, George Mason University, Fairfax, VA, 1989.

Kerschberg, L. (ed.), Expert Database Systems: *Proceedings from the First International Workshop*, Benjamin/Cummings Publishing Company, Menlo Park, CA, 1986.

Kerschberg, L. (ed.), Expert Database Systems: *Proceedings from the First International Conference*, Benjamin/Cummings Publishing Company, Menlo Park, CA, 1987.

Kerschberg, L. (ed.), Expert Database Systems: *Proceedings from the Second International Conference*, George Mason University, Fairfax, VA, Benjamin/Cummings Publishing Company, Menlo Park, CA, 1988.

Kokar, M.M., "Coper: A Methodology for Learning Invariant Functional Descriptions," in *Machine Learning: A Guide to Current Research*, Michalski, R.S., Mitchell, T.M and Carbonell, J.G. (Eds.), Kluwer Academic Publishers, Boston, MA 1986.

Langley, P., Bradshaw G.L. and Simon, H.A., "Rediscovering Chemistry with the BACON System," in *Machine Learning: An Artificial Intelligence Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (Eds.), Morgan Kaufmann, San Mateo, CA, 1983.

Layman, T.C., "A PASCAL Program to Convert Extended Entry Decision Tables into Optimal Decision Trees," Department of Computer Science, Internal Report, University of Illinois, Urbana, IL, 1979.

Michalski, R.S., "Designing Extended Entry Decision Tables and Optimal Decision Trees Using Decision Diagrams," Report No. UIUCDCS-R-78-898, Department of Computer Science, University of Illinois, Urbana IL, 1978.

Michalski, R.S., "Theory and Methodology of Inductive Learning," in *Machine Learning: An Artificial Intelligence Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (Eds.), Morgan Kaufmann, San Mateo, CA, 1983.

Michalski, R.S., "Toward a Unified Theory of Learning: Multistrategy Task-adaptive Learning," in *Reports of Machine Learning and Inference Laboratory*, MLI 90-1, Center for Artificial Intelligence, George Mason University, Fairfax, VA, 1990.

Michalski R.S. and Baskin, A.B., "Integrating Multiple Knowledge Representations and Learning Capabilities in an Expert System: The ADVISE System," *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, pp. 256-258, 1983.

Michalski, R.S., Baskin, A.B. and Spackman, K.A., "A Logic-based Approach to Conceptual Database Analysis," *Sixth Annual Symposium on Computer Applications in Medical Care* (SCAMC-6), George Washington University Medical Center, Washington, DC, pp. 792-796, 1982.

Michalski, R.S., Baskin, A.B., Uhrik, C. and Channic, T., "The ADVISE.1 Meta-Expert System: The General Design and a Technical Description", Report No. UIUCDCS-F-87-962, Department of Computer Science, University of Illinois, Urbana, IL, 1987.

Michalski, R.S. and Chilausky, R.L.,"Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis", *International Journal of Policy Analysis and Information Systems*, Vol. 4, No.2,1980.

Michalski, R.S., Davis,J.H., Bisht,V.S. and Sinclair, J.B.,"PLANT/DS: An Expert Consulting System for the Diagnosis of Soybean Diseases, "*Plant Diseases* and Proceedings of the First European Conference on Artificial Intelligence, Orsay, France, July 12-14, pp. 133-138, 1982.

Michalski, R.S., Davis, J.H., Bisht, V.S. and Sinclair, J.B., "A Computer-Based Advisory System for Diagnosing Soybean Diseases" in *Illinois Plant Disease*, pp. 459-463, April 1983.

Michalski, R.S., Ko, H. and Chen, K., "SPARC/E(V.2), An Eleusis Rule Generator and Game Player," ISG 85-11, UIUCDCS-F-85-941, Department of Computer Science, University of Illinois, Urbana, IL, February 1985.

Michalski, R.S., Ko, H. and Chen, K., "Qualitative Process Prediction: A Method and Program SPARC/G," *Expert Systems*, C. Guetler, (Ed.), Academic Press Inc., London, 1986.

Michalski, R.S., Kerschberg, L. Kaufman. K. and Ribeiro, J.,"Mining for Knowledge in Databases: The INLEN Architecture, Initial Implementation and First Results" invited paper for the *International Journal on Intelligent Systems*, No. 1, 1992.

Michalski R.S. and Larson, J.B., "Selection of Most Representative Training Examples and Incremental Generation of VL_1 Hypotheses: the underlying methodology and the description of programs ESEL and AQ11," Report No. 867, Department of Computer Science, University of Illinois, Urbana, IL, 1978.

Michalski, R.S. and Larson, J.B., rev. by Chen, K., "Incremental Generation of VL1 Hypotheses: The Underlying Methodology and the Description of the Program AQ11," Report No. UIUCDCS-F-83-905, Department of Computer Science, University of Illinois, Urbana, IL, 1983.

Michalski, R.S., Mozetic, I., Hong, J. and Lavrac, N., "The AQ15 Inductive Learning System: An Overview and Experiments," Report No. UIUCDCS-R-86-1260, Department of Computer Science, University of Illinois, Urbana, IL, 1986.

Michalski R.S., and Stepp, R.E., "Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy," IEEE Transactions on Pattern Analysis and Machine Intelligence, 1983.

Michalski, R.S., Stepp, R.E. and Diday, E., "A Recent Advance in Data Analysis: Clustering Objects into Classes Characterized by Conjunctive Concepts," in *Progress in Pattern Recognition*, Vol. 1, L. N. Kanall and A. Rosenfeld (Eds.), New York: North-Holland, pp. 33-56, 1981.

Mozetic, I., "Compression of the ECG Knowledge-base Using the AQ Inductive Learning Algorithm," ISG 85-13, UIUCDCS-F-85-943, Department of Computer Science, University of Illinois, Urbana, March 1985.

Newkirk,P., "TURF: An Expert System for Ground Development," ISG 85-3, UIUCDCS-F-85-933, Department of Computer Science, University of Illinois,Urbana, January 1985.

Quinlan, J.R., "Probabilistic Decision Trees," in *Machine Learning: An Artificial Intelligence Approach, Volume III*, Y. Kodratoff and R.S. Michalski, (Eds.), Morgan Kaufmann, San Mateo, CA, 1990.

Reinke, R., "PLANT/DS: An Expert System for the Diagnosis of Soybean Diseases Common in Illinois, User's Guide and Program Description," ISG 83-12, UIUCDCS-F-83-912, Department of Computer Science, University of Illinois, Urbana, October 1983.

Reinke, R.E., "Knowledge Acquisition and Refinement Tools for the ADVISE Meta-Expert System," Master's Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1984.

Spackman, K.A., "QUIN: Integration of Inferential Operators within a Relational Database," ISG 83-13, UIUCDCS-F-83-917, M. S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1983.

Stepp, R.E., "Learning without Negative Examples via Variable-Valued Logic Characterizations: The Uniclass Inductive Program AQ7UN1," Report No. 982, Department of Computer Science, University of Illinois, Urbana, IL, 1979.

Stepp, R.E., "A Description and User's Guide for CLUSTER/2, a Program for Conceptual Clustering," *Reports of the Department of Computer Science*, University of Illinois, Urbana, IL, 1983.

Stepp, R.E., "Conjunctive Conceptual Clustering: A Methodology and Experimentation," PhD Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1984.

Uhrik,C.T., "PLANT/DS Revisited: Non-Homogeneous Evaluation Schema in Expert Systems," Proc. of the National Conference on Artificial Intelligence, AAAI-82, Pittsburgh, August 18-20, 1982.

Wnek, J. and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17: A Method and Experiments," in *IJCAI-91 Workshop on Evaluating and Changing Representations in Machine Learning*, Sydney, Australia, 1991.

Wnek, J. and Michalski, R.S., "An Experimental Comparison of Symbolic and Subsymbolic Learning Paradigms: Phase I -- Learning Logic-Style Concepts," in *Proceedings of the First International Workshop on Multistrategy Learning*, Harpers Ferry, WV, pp. 324-339, 1991.

Wnek, J., Sarma, J., Wahab, A. and Michalski, R.S., "Comparing Learning Paradigms via Diagrammatic Visualization," in *Methodologies for Intelligent Systems 5*, Ras, Z., Zemankova, M., Emrich, M. (Eds.), Elsevier Science Publishers, New York, pp. 428-437, 1990.

Zhang, J., "Integrating Symbolic and Subsymbolic Approaches: Learning Flexible Concepts," to appear in Michalski, R.S. and Tecuci, G. (Eds.), *Machine Learning: A Multistrategy Approach*, Volume IV, Morgan Kaufmann, San Mateo, CA, 1992.

Zytkow, J.M., "Combining Many Searches in the FAHRENHEIT Discovery System," in *Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, CA, 1987, pp. 281-287.