MINING FOR KNOWLEDGE IN DATABASES: THE INLEN ARCHITECTURE, INITIAL IMPLEMENTATION AND FIRST RESULTS

by

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

Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies, Vol. 1, No. 1, pp. 85-113, August 1992.

Journal of Intelligent Information Systems, 1, 85–113 (1992)
© 1992 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Mining for Knowledge in Databases: The INLEN Architecture, Initial Implementation and First Results

R.S. MICHALSKI, L. KERSCHBERG AND K.A. KAUFMAN
Center for Artificial Intelligence, George Mason University, Fairfax, VA 22030

J.S. RIBEIRO
The Analytic Sciences Corporation, 12100 Sunset Hills Road, Reston, VA 22090 and Center for Artificial Intelligence, George Mason University, Fairfax, VA 22030

Abstract. The architecture of an intelligent multistrategy assistant for knowledge discovery from facts, INLEN, is described and illustrated by an exploratory application. INLEN integrates a database, a knowledge base, and machine learning methods within a uniform user-oriented framework. A variety of machine learning programs are incorporated into the system to serve as high-level *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 rules from facts, 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, deriving statistical properties and using them for qualitative evaluations, etc. The initial implementation of the system, INLEN 1b, is described, and its performance is illustrated by applying it to a database of scientific publications.

Keywords: knowledge discovery, machine learning, databases, multistrategy systems

1. 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 industry, military, government or science, thousands of databases have been developed to capture information relevant to some particular class of tasks.

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 anlaysis can detect a correlation between given factors, but does not produce a conceptual explanation why such a correlation exists, nor does it formulate any specific quantitative and/or qualitative law(s) responsible for this correlation. A statistical technique can determine a central tendency and variability of some properties, or fit a curve to a set of datapoints, but it cannot explain them in terms of causal

dependencies or qualitative relationships. A numerical taxonomy technique can create a classification of entities and determine a numerical similarity among the entities assembled into the same or different classes, but it will not create a qualitative description 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 these limitations by applying modern techniques of machine learning and discovery to complex databases. Specifically, this paper describes the goals, 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 **in**ference and **learn**ing, 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 existing data and knowledge. This is accomplished through various forms of inference, including deduction, induction and analogy. The INLEN system is designed to ultimately include a wide spectrum of knowledge generation operators, such as those for creating conceptual descriptions of sets of facts, identifying logical regularities and similarities among facts or groups of facts, inventing conceptual classifications of data, generating new attributes to better describe data, selecting relevant examples or attributes, formulating equations governing quantitative data as well as the conditions of their applicability, as well as operators representing known numerical and statistical techniques. These operators employ all basic forms of inference—deduction, induction or analogy. Thus, they allow a user to explore the data using a variety of strategies, and therefore INLEN can be viewed as a multistrategy data exploration system.

The emphasis of this paper is to present the architecture of the INLEN system, in particular, to demonstrate how combining various programs and techniques in such an environment can lead to a powerful, general-purpose knowledge-discovery system. We will explore some of the ways in which this architecture may be applied.

The motivating idea behind the INLEN system is to integrate the three technologies in order to provide a user with a powerful tool for manipulating both data and knowledge, and for extracting from that data and/or knowledge new or better knowledge. The INLEN approach is to build an "intelligent assistant" that could ultimately improve the effectiveness of a data analysis expert, and obtain important discoveries and conclusions partially on its own. Such a system would be an assistant for the conceptual analysis, discovery, and explanation of patterns in databases. It would be equipped with heuristics characterizing the type of information or a class of patterns that might be "important" to a

user. It would also provide a variety of advanced machine learning and plausible reasoning techniques for implementing a search for such information or patterns.

The role of such an intelligent assistant would be 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., [29]), or discover regularities that would be hard to formulate without the aid of a program (e.g., [13]). 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. A learning program can be especially helpful in such situations. It can also help to avoid the possible human error of overlooking something of note in the data. A system that could process and filter data faster than a human analyst, with an equal or lower error rate, would be very useful in domains where large amounts of data have to be analyzed.

The approach that we are developing 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. Working together, such a human-computer data analysis system could exhibit a synergistic effect in extracting useful knowledge from data, and have an increased potential for making discoveries. A machine learning system might also be potentially useful in formulating explicit criteria that experts are using implicitly in evaluating the "interestingness" of a pattern.

Systems able to extract useful knowledge from large databases could be useful in many fields, such as complex decision making, resource allocation and management, business transactions, 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.

The INLEN system will enable a user to apply advanced machine learning and plausible reasoning tools for determining symbolic, rather than numeric, descriptions of data, discovering high level regularities, and proposing conceptual explanations of them. The system will be capable of using techniques of the theory of human plausible reasoning [8] to hypothesize missing values in a database, or to propose the most plausible data interpretations.

The key factor in the development of an expert system is building its knowledge base. Current methods for that purpose are complex, time-consuming and error-prone (e.g., [4]–[6]). Most of the difficulties come from the fact that the system lacks the capability of self-improving its knowledge through experience. It is believed that modern machine learning methods (e.g., [25], [43]) can be used for partially automating knowledge acquisition and evolution. For instance, by using empirical induction, a system can learn general concepts or rules characterizing a set of examples. By using analogical learning, a system may acquire knowledge about an unknown entity by modifying prior knowledge about a similar entity. By using explanation-based learning, a system may transform inefficient knowledge into efficient rules. Until now these single-strategy learning methods have not made a significant impact in the field of knowledge acquisition because each strategy requires specific conditions in order to be applicable. INLEN integrates these basic learning strategies in a goal-directed manner, and utilizes these combined strategies for improved knowledge acquisition.

The underlying data and knowledge representation in INLEN employ *knowledge segments*, which link relational tables with rules, equations and/or hierarchies. A 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 (Query and Inference), a combined database management and data analysis environment [31], [32], [44]. QUIN was designed both as a stand-alone system, and as a subsystem of ADVISE, a large-scale inference system for designing expert systems [2], [31], [33]. In the last few years, new tools have been developed; in particular, more advanced inductive learning systems, e.g., AQ15 [38] and ABACUS-2 [14], and expert database systems [21]–[23]. The above systems have influenced the development of INLEN. INLEN also draws upon the experiences with AGASSISTANT, a shell for developing agricultural expert systems [17], AURORA, a general-purpose PC-based expert system shell with learning and discovery capabilities, designed by Michalski and Katz [16], and EMERALD [19], [20], a user-oriented multiprogram environment for education and research in machine learning.

In Section 2, we present an example that demonstrates some of the knowledge discovery tasks that a data analyst may need to perform. Section 3 discusses in depth the architecture of INLEN, and how it addresses the problems brought up in Section 2. Section 4 focuses on the implementation of INLEN-1. Section 5 describes the results of applying INLEN-1 to a database containing information on scientific publications from the (then) Soviet Union. In Section 6, we summarize this paper and discuss issues for future research.

The key factor in the development of an expert system is building its knowledge base. Current methods for that purpose are complex, time-consuming and error-prone (e.g., [4]–[6]). Most of the difficulties come from the fact that the system lacks the capability of self-improving its knowledge through experience. It is believed that modern machine learning methods (e.g., [25], [43]) can be used for partially automating knowledge acquisition and evolution. For instance, by using empirical induction, a system can learn general concepts or rules characterizing a set of examples. By using analogical learning, a system may acquire knowledge about an unknown entity by modifying prior knowledge about a similar entity. By using explanation-based learning, a system may transform inefficient knowledge into efficient rules. Until now these single-strategy learning methods have not made a significant impact in the field of knowledge acquisition because each strategy requires specific conditions in order to be applicable. INLEN integrates these basic learning strategies in a goal-directed manner, and utilizes these combined strategies for improved knowledge acquisition.

The underlying data and knowledge representation in INLEN employ *knowledge segments*, which link relational tables with rules, equations and/or hierarchies. A 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 (Query and Inference), a combined database management and data analysis environment [31], [32], [44]. QUIN was designed both as a stand-alone system, and as a subsystem of ADVISE, a large-scale inference system for designing expert systems [2], [31], [33]. In the last few years, new tools have been developed; in particular, more advanced inductive learning systems, e.g., AQ15 [38] and ABACUS-2 [14], and expert database systems [21]–[23]. The above systems have influenced the development of INLEN. INLEN also draws upon the experiences with AGASSISTANT, a shell for developing agricultural expert systems [17], AURORA, a general-purpose PC-based expert system shell with learning and discovery capabilities, designed by Michalski and Katz [16], and EMERALD [19], [20], a user-oriented multiprogram environment for education and research in machine learning.

In Section 2, we present an example that demonstrates some of the knowledge discovery tasks that a data analyst may need to perform. Section 3 discusses in depth the architecture of INLEN, and how it addresses the problems brought up in Section 2. Section 4 focuses on the implementation of INLEN-1. Section 5 describes the results of applying INLEN-1 to a database containing information on scientific publications from the (then) Soviet Union. In Section 6, we summarize this paper and discuss issues for future research.

2. A motivational example

Consider a relational database, i.e., a collection of n-ary relations represented as a collection of tables whose rows (tuples) represent different events or examples, and whose columns represent the various attributes (or variables) that characterize each object. We assume the possibility that some locations in a table may be marked "?" to indicate unknown values, or marked "N/A" denoting the nonapplicability of some attributes to some objects. For example, in the sample relational table depicted in Table 1, some of the trains do not have conductors, the experimental nuclear monorail (#6) lacks cars and loads, and does not run on any regular train line, nor does the government military supply transport (#19).

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

		Attributes								
	Train #	# Cars	Load	Fuel	Eng Wls	Condctr	Engineer	CarTyp	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
T	10	37	Cars	Diesel	18	Joan	Steve	Flat	Penn	West
u	11	33	CD's	Electric	4	Dale	JJ	J-top	Rio	West
p	15	9	Passngr	Coal	8	Bill	Ashraf	Flat	ICG	South
l	19	83	MilSupp	?	12	N/A	?	Tank	N/A	?
e	20	6	Fuel	Coal	6	Janet	Lou	Coal	NCRR	North
s	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

We will also assume that encoded in the system is rudimentary background

knowledge about the attributes that characterize the data, the values they can have, and the relationships among those values (linear, hierarchical, etc.), and possibly among the attributes. For example, a rule about the variables in this data set might be: ***Conductor*** = *N/A if* **# *Cars*** = *0*. In general, we can talk about the functional and multivalued dependencies within the data.

Given such a table, a data analyst will try to discover useful facts about the objects represented in the table. For example, if the analyst wanted to find rules for determining the direction in which a train was traveling, one set of rules that might be discovered would be as follows:

Direction is North if **EngWls** = 6 to 10
Direction is South if (**Fuel** = Electric and **# Cars** < 7) or (**Fuel** = Coal and **CarTyp** = Flat)
Direction is East if **Load** ≠ Animals and **Line** = B&O or C&O or Metro or SP or MP or Conrail
Direction is West if **# Cars** > 16
Direction is unknown if **Fuel** = Nuclear or **Load** = MilSupp

But suppose it were too difficult to ascertain the fuel used by a train. The analyst would like to create a rule that used other attributes in determining direction such as:

Direction is South if (**Eng Wls** ≥ 8 and **# Cars** ≤ 6) or (**# Cars** ≤ 9 and **CarTyp** = Flat)

Perhaps the analyst wished to see ways of grouping the trains other than by direction. By using a conceptual clustering operator, other groupings could be created such as:

A train is in **Class 1** if **CarTyp** is open
A train is in **Class 2** if **CarTyp** is closed
A train is in **Class 3** if **CarTyp** is unknown or N/A

The analyst might wish to reduce the data table, either by removing less useful attributes or less representative instances. Or the analyst might want to enhance the table by predicting unknown values, for instance guessing that the most likely value for the car type of train 7 is a tank car. The table could also be expanded by generating plausible examples of, for example, westbound trains with over 10 cars.

In general, there is potentially a wide variety of tasks that make up the analysis of a set of data. INLEN is designed to be capable of handling many of these tasks by integrating different types of knowledge discovery programs as individual operators in a global environment.

3. The architecture of INLEN

As mentioned above, INLEN combines database, knowledge base and machine learning capabilities in a single large-scale system. It includes ideas from the recently developed expert database technology to combine the storage and access capabilities of a database system with the ability to derive well-founded conclusions from a knowledge-based system [21]–[23]. INLEN integrates several advanced machine learning capabilities, which until now have existed only as separate experimental programs. Many learning systems are capable of a narrow subset of the spectrum of knowledge that can be gained from factual data. By integrating a variety of these tools, a user will have access to a very powerful and versatile system.

The general design of INLEN is shown in Figure 1, and expands upon the initial design presented in [18]. The most important new features include several novel knowledge generation operators and the idea of macro operators and data analysis programs, while the latest design groups the individual machine learning operators into higher-level operator classes based on their functions. As is depicted in Figure 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 and/or knowledge base. This knowledge base can contain not only knowledge about the contents of the database, 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 and extract useful knowledge from a database, 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 learning. 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).

Information in the database consists of relational tables (RTs), and information in the knowledge base consists of units called *knowledge segments*. A knowledge segment (KS) can be simple or compound. Simple KSs include rulesets, equations, networks and hierarchies. Compound KSs consist of combinations of any of the above, or combinations of KSs and RTs. In this way, knowledge may be associated with a group of relations, attributes, domains, or other knowledge to which it applies. Thus, we can know the confidence we can attribute to a

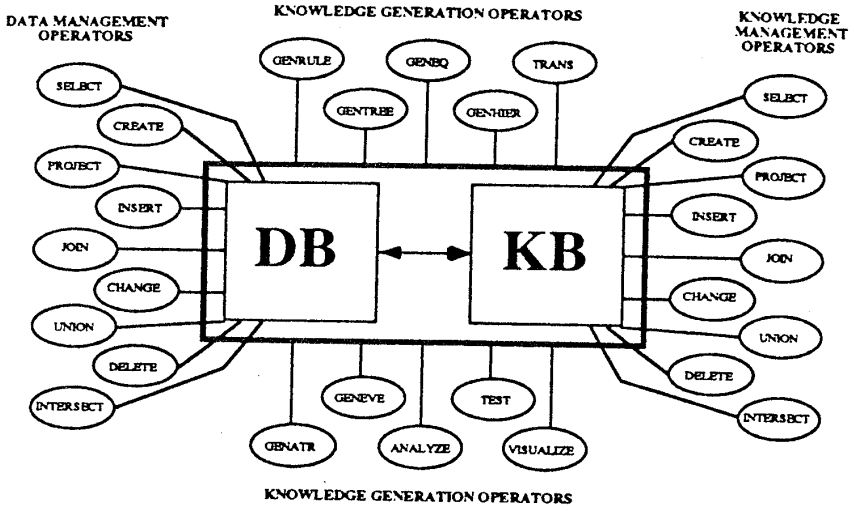

Fig. 1. A top-level functional architecture of INLEN.

piece of discovered knowledge and know how the knowledge may be used. The latter form may 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 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 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.

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 database. Thus, they operate on RTs. The individual operators are described in Section 3.1.1. The knowledge management operators perform analogous tasks on the knowledge base, in situations in which manual input, access or adjustments are required, and are catalogued in Section 3.1.2. The knowledge generation operators, on the other hand, interact with both the database and the knowledge base. They

take input from both the database and the knowledge base, and 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 knowledge segments. Figure 1 shows the 10 major classes of KGOs. These classes and their individual members will be discussed in detail in Section 3.1.3. The macro operators differ from the other operators in that they consist of sequences of the other operators, along with instructions to control the flow of commands within the macro. For more complex operations, programs can be written to direct discovery in INLEN. Both macro-operators and data analysis programs are discussed in Section 3.1.4.

3.1. Description of operators

A brief description of each of the DMOs, KMOs and KGOs, and a further description of the macro operators follows.

*3.1.1. **Data management operators (DMOs).*** The data management operators perform a standard set of relational database operations for the purpose of manipulating the system's collection of facts. Their inputs may be tuples, tables, and/or specifications, and their outputs will generally consist of new relational tables. The DMOs are listed here for the sake of completeness.

CREATE generates a new relational table. It takes an attribute list and the name of the new table as arguments, and produces an empty relational table.
INSERT adds a new tuple (row) to an already existing relational table.
CHANGE alters some or all of the values in some or all of the tuples of a table, and returns the modified table.
DELETE removes rows from a table, or columns from its specification, as specified respectively by SELECT or PROJECT operations, returning the reduced table. Alternatively, entire tables may 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 outputs a subtable of the input table by removing columns and then any duplicate tuples. Columns that are retained 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) which appear in either of the two tables.

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

3.1.2. Knowledge management operators (KMOs). The knowledge management operators 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 may take the form of simple or compound knowledge segments (KSs). Consequently, most of the knowledge management operators shown in Figure 1 are generalized for any of these forms. Unless otherwise specified, they should be thought of as operating on any KS, i.e., they can operate on rules, equations, hierarchies, etc.

The diverse representations of knowledge may be culled from the *same* database, and will therefore represent distinct viewpoints obtained using the knowledge generation operators. For example, a dynamical 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. The 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, etc. Each of these viewpoints is valid, and should be managed by the KMOs.

Expert database tools and techniques can be used to manage the evolution of the combined knowledge/database by incorporating knowledge discovered in the database. The arrow in Figure 1 linking the DB and the KB components represents such an interaction.

The knowledge management operators listed below are depicted as analogs of INLEN's data management operators. Without intensive testing of the system in different domains, one cannot determine how useful these operators are, but they represent our first approximation based on the analogy to the data management operators. The KMOs may be expanded to permit specific knowledge acquisition techniques, as discussed in [4]–[6]. Further research may lead to the development of other operators, and also other knowledge representations, including the likely use of an object-oriented approach in which one data representation is replaced by an active link to the discovered concept, which may be a formula, a ruleset, or some other appropriate representation. Under INLEN's design, these are the knowledge management operators and their functions:

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

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

CHANGE is used for the manual alteration of part of one or more items in a knowledge segment.

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

SELECT is used to retrieve a knowledge segment from the knowledge base (and from the database in the case of component RTs). Criteria may be provided to return only selected items (such as rules, subtrees, rows in tables, etc.) in the retrieved KS.

PROJECT is used to return a subset of a compound KS which ignores entire components (e.g., rulesets, decision trees, columns of tables) of the KS. The items specified in the operator's arguments will be included.

JOIN is used to combine a pair of simple knowledge segments or components of compound knowledge segments. For example, a set of rules and a data table can be united into a compound KS, or two rulesets may be combined by finding conditions in the first ruleset which are satisfied by decisions in the second ruleset. Rules may then be expanded by replacing the matching conditions in the first ruleset with the conditions leading to the corresponding decisions in the second ruleset.

UNION is applied to two or more knowledge segments 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 knowledge segments of the same type. It generates a list of only those elements present at least once in each of the segments.

3.1.3. Knowledge generation operators (KGOs). 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 will generally be compound KSs which include indexing tables within the knowledge base.

Each of the knowledge generation operators requires certain background knowledge and parameters. The background knowledge consists of information about the domain, the variables in the data tables, etc. The parameters include specifications for choosing an output description from multiple possibilities. For simplicity, these inputs are not included in the following descriptions, as they are assumed to be part of any KGO.

In general, a KGO takes one table from the database and one or more knowledge components from the knowledge base, and generates one or more new tables and/or knowledge segments. This knowledge discovery may use either an incremental or a batch mode; in the former case, the emphasis is on improving, refining, or adjusting the strength of existing knowledge, while in the latter case, wholly new knowledge is being discovered from facts and/or other knowledge.

It is practical to group the KGOs by the types of output they generate (see Figure 1). Those whose primary output consists of rules are separated from those that generate tuples, for example. We discuss each of the KGOs, sorted by

the output group to which they belong. Most of these operators are extensions of existing programs; while this section describes their tasks, the programs on which they are based will be mentioned in Section 4.

GENRULE: Generate rules. Operators in the GENRULE class take some form of 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.

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.

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.

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.

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.

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, logical conditions of their applicability and the elements of the input table to which they apply.

GENHIER: Generate hierarchies. The GENHIER operators conceptually classify an input set of tuples, rules, equations, etc. The CLUSTER operator creates a logical division of the input objects into two or more groups (a hierarchy one level deep), while the TAXONOMY operator generates a full-fledged classification hierarchy, and can be viewed as 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.

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 giving it new examples to learn from. 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 can take a fact such as "The Lincoln Town Car is a luxury automobile" as input, and create 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 set size covered by the input KS. The rule, "A creature is in class 1 if it is a dog," can be expanded to "A creature is in class 1 if it is a mammal" using the GENERALIZE operator, or constrained to "a creature is in class 1 if it is a golden retriever" via the SPECIALIZE operator.

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.

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 Attribute) 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 those 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 mathematical operators specified in its arguments in order to combine variables into useful composites. The output from this operator 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 can be used if the sum or product of two variables might be more useful than either individual value.

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

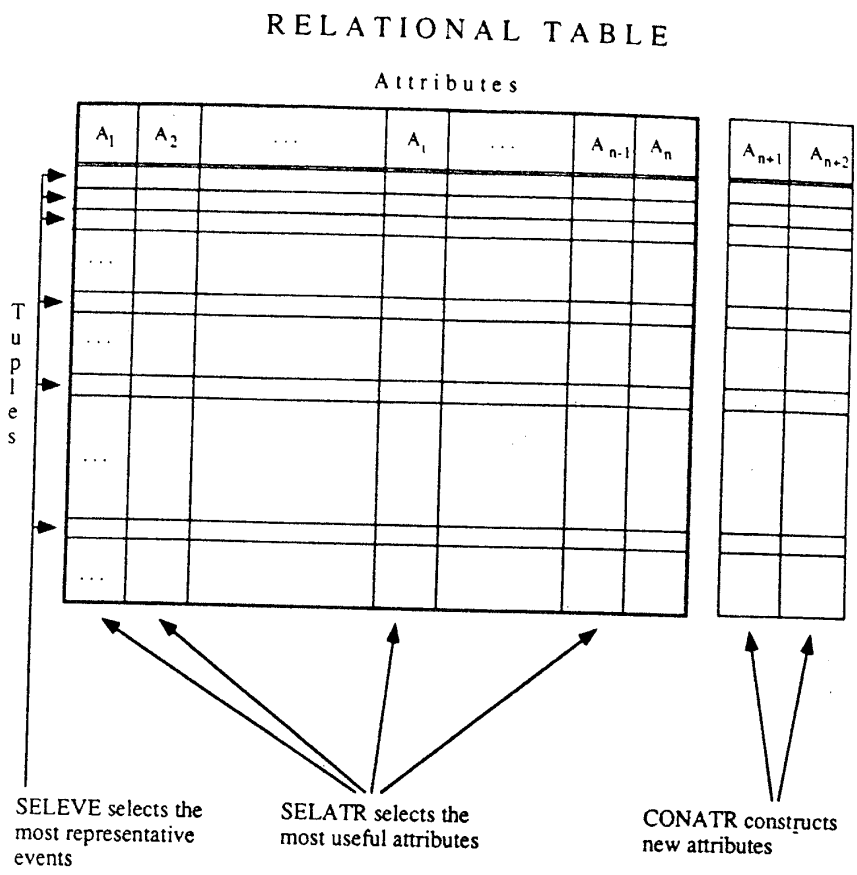

Fig. 2. An illustration of the roles of the SELEVE, SELATR, and CONATR operators.

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, 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.

ANALYZE: Analyze data. The ANALYZE family of operators return knowledge in the form of numerical weights that describe the elements in the database. These numbers can represent 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 will be a knowledge segment that links the inputs with any discovered relationships, and includes a numerical representation of the strength of the relationships.

RELEVE (Relate Events) behaves similarly to RELATR, but it instead determines relationships among elements 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 will again consist 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 of the data in order to determine its various statistical properties. 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 rule-set 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 two different rules. 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 weights that indicate the quality of the knowledge segments. The primary output table is in the form of a *confusion matrix*, i.e., a matrix whose (i, j)th element shows how many examples from the class i were classified by the rules to be in class j. TEST also creates links between the output tables and the input structures.

VISUALIZE: Diagrammatic visualization. VISUALIZE displays a set of data, as specified by rules or transformations of it, graphically on the screen. Input to this operator can be a relational table, or a knowledge segment characterizing part of a relational table's event space. For example, one might invoke VISUALIZE with the descriptor "employees that are female, between ages 35 and 50, and have over five years of seniority." The output from this operator appears as a two-dimensional representation of the event space, with the input set highlighted. Figure 3 illustrates the DIAV program [49] on which this operator is based, using as a domain a collection of robots, whose identifying features include head shape, jacket color, what they are holding, and whether or not they are smiling.

3.1.4. Macro operators and data analysis programs. During data analysis, a user may apply one operator at a time, or may want to apply a sequence of operators to a given dataset. Depending on the data analysis task, different sequences of operators may have to be applied. It is possible that some of these sequences will be repeated often. In order to facilitate the execution of such operator sequences, INLEN is designed to provide mechanisms for creating macro operators and high-level data analysis programs.

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 macro operator 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

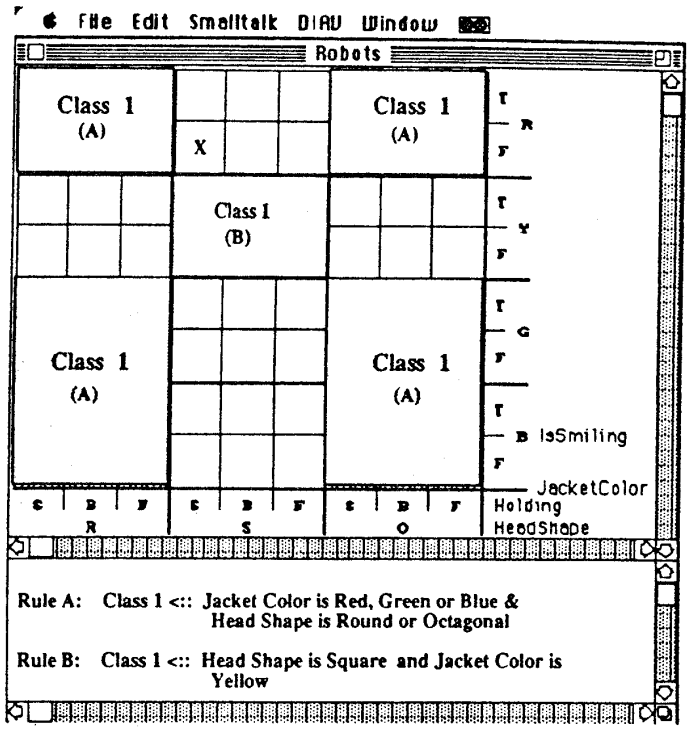

Fig. 3. An example of the diagrammatic visualization of a concept by the VISUALIZE operator.

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 database 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.

4. The implementation of INLEN-1

4.1. Implementation plan

INLEN is a large-scale system composed of many modules, some of which can serve as powerful stand-alone systems. In order to implement such a large

system with broad capabilities, we have undertaken its development in stages, which represent versions with an increasing set of capabilities. The design and implementation of the system builds upon the development of the QUIN [31], [32], [44], ADVISE [2] and AURORA [16] systems.

The first stage of development (version INLEN-1) includes a knowledge base of simple decision rules, a relational database, and an extensive user-oriented menu-based graphical interface. The knowledge generation operators (KGOs) include such operators as: CHARSET, DIFFSET, IMPROVE, TEST, and PREDVAL. The system has been initially implemented on an IBM-compatible computer. We have tested it on a variety of problems requiring rule discovery from facts or improvement of existing rules. INLEN-1 can be divided into three levels of development, with the first two systems currently operational. The first, INLEN-1a, was closely tied to the AURORA system, running in a limited Pascal environment. INLEN-1b is a C-based version of INLEN-1a that runs more efficiently than its predecessor and has the capacity for larger data and knowledge bases. The development of INLEN-1c calls for a standardization and restructuring of the code and the addition of several features.

The second major phase of INLEN development includes the creation of a larger prototype on a Sun workstation that integrates a full-fledged knowledge base with a commercial-grade database. The system includes most of the knowledge generation operators indicated in Section 2.1.3, and a new system interface.

The third stage will involve the development and implementation of the remaining operators, and modifications to the structure of the system's components based on the results generated during the previous stage. Research associated with this stage will include the development of a control system that will allow INLEN to make some autonomous control decisions and operator selection based on the knowledge it already has acquired.

Many of the INLEN operators are based on the research results and programs developed over the last 15 years at this 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. The latter ones include the CONEVE, PREDVAL, RELATR, RELEVE, and RELKS operators.

We are in the process of designing the user interface and its links to the DMOs and KMOs for the more advanced versions of INLEN. As mentioned before, many of the KGOs are based on programs and/or methods developed in the past. Here is a brief summary of the origin and predecessors for the set of KGOs.

CHARSET's is based on research on learning characteristic descriptions from examples. An early program for this purpose, UNICLASS, is described in [45].

A version of the program that executes both CHARSET and DIFFSET, AQ11, is described in [37]. 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 combined with the TRUNC method for two-tiered concept representation [15]. More recent research has resulted in the incorporation of further capabilities into the AQ system, such as learning flexible concepts [51], [52], and capabilities for constructive induction [3], [48].

CHARSEQ and DIFFSEQ represent an extension of the SPARC methodology for determining patterns in sequences, described in [11], [34], [35]. 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.

EVENTREE is based on ideas implemented in the C4.5 program for generating decision trees from examples [41] and ASSISTANT [7]. The RULETREE operator utilizes the OPTTREE program for creating decision trees from rules [27], [28]. Both are ready for implementation within INLEN.

GENEQ is based on the ABACUS-2 system for integrated qualitative and quantitative discovery [14], an extension of ABACUS [13]. These quantitative discovery programs are related to systems such as BACON [26], FAHRENHEIT [53], and COPER [24].

CLUSTER and TAXONOMY use the conceptual clustering algorithm from the CLUSTER program described in [39], [40], [46], [47]. This program is operational at present.

The methods used by the ABSTRACT and CONCRETIZE operators have not been heavily studied in the past. [30] discusses the place of abstraction and concretion within multistrategy learning environments, and further research into this area has begun. An additional topic of current research is the development of intelligent control strategies for the GENERALIZE and SPECIALIZE operators.

The methodology for IMPROVE is based on the AQ15 program [15]. AQ15 has been implemented and tested in conjunction with other environments.

The GENATR operators are based on existing programs. SELATR employs

the VARSEL algorithm [1]. CONATR incorporates the capabilities of the CONVART program [10].

SELEVE uses the ESEL methodology, described in [9], [36]. CONEVE and PREDVAL are the subject of research to build upon the foundations set by the performance elements of the SPARC and APPLAUSE [12] programs.

SIMILIZE is being developed from an application of object characterization programs such as AQ.

RELATR, RELEVE, and RELKS are subjects of current research, while GENSTAT will incorporate existing statistical packages to generate its output.

TEST uses the ATEST methodology [42] for analyzing consistency and completeness in rules, and generating confusion matrices.

VISUALIZE uses the DIAV diagrammatic visualization methodology, currently being developed [49], [50].

4.2. Features of INLEN-1

INLEN-1 consists of six major functions:

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

In the application system definition module, the user may define the database schema, import facts into the database, and specify parameters that will guide the behavior of the learning and advisory modules. Attributes in the database may be defined as being nominal, linear, or hierarchically structured. The knowledge acquisition module consists of programs that are able to support both the direct entry of rules and the creation of a knowledge base via the analysis of facts. The learning and discovery module contains programs for rule learning from examples, rule optimization, rule improvement through examples and rule testing for completeness and consistency.

The advisory and prediction module extrapolates the values of unknown attributes in the data through a three-stage process. In the first stage, *reduction*, the set of possible hypotheses is reduced by testing them against facts about other attributes critical to the value of the attribute being investigated (these attributes are identified during the definition of the database schema). During the second stage, *discrimination*, the remaining hypotheses are scrutinized further in order to generate a most likely value for the attribute in question. The third

stage, *confirmation*, involves further scrutiny of this leading candidate hypothesis in order to either confirm it (through its confidence value either exceeding a user-defined threshold or exceeding those of all other hypotheses under consideration by some preset amount) or disconfirm it (through another hypothesis' confidence value exceeding that of the leading candidate).

The system review module consists of utilities to allow the user to view the contents of the data and knowledge bases, the database schema, and the learning and inference parameters.

5. Experimental application: Discovering patterns in a database of scientific publications

One experimental domain which we have explored involves a database of scientific publications written by scientists in the Commonwealth of Independent States (CIS). This database (which we refer to as the CISA database) contains 2841 records, where each record contains information on a particular paper published by a CIS author. Some of the records contain values for all fields while other records contain some missing or nonapplicable data.

5.1. Database definition

Currently, INLEN 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 which we created, every record lists the attributes and the corresponding values for each publication. Attributes of the CISA database that were included in our analysis are

1. AUTHOR—Author's name.
2. COAUTHOR—Coauthors' name.
3. TITLE—Title of the paper.
4. PUBYR—Publication year of the paper.
5. INSTITUTE—Research institute 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 database:

1. AUTHOR = Aleksandrov, Yu 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.

5.2. *Experiments and results*

The goal of this experimental application was to learn as much as we could about the CIS institutes from the data we had available. A secondary goal was to use the acquired knowledge to fill in any missing data related to the INSTITUTE field. This knowledge would be expressed in terms of relationships between the INSTITUTE field and the other attributes in the CISA database. 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 database with 179 records. Using this view, we then generated rules for each institute in our database. 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.

This analysis led to several interesting discoveries about the CIS institutes. 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 discovery was a rule which related technical specialty areas with the Kaunas Polytechnical Institute. This rule was created with the same knowledge creation parameters; the actual rule produced by INLEN in this case was

INSTITUTE is Kaunas Polytechnical Institute if SPECIALTY2 is Wideband c Satellites.

This rule illustrates the presence of disjuncts to form a simple rule and tell us that research on wideband satellites occurs at this institute. These result illustrate just two of the interesting relationships discovered from the CISA database using INLEN.

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 database 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 findings:

- The system allows users without a large amount of domain expertise to discover interesting relationships in the data.
- It can discover relatively simple patterns that have been hidden by the volume of the data.
- It may expose 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 database. 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 databases. Finally, the system should have an ability for supporting an evolution of the database and knowledge base schema to be able to adequately incorporate new data and learned knowledge.

6. Conclusion 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 in data analysis and discover interesting regularities in them. These regularities can be detected in qualitative

data, quantitative data, and in the knowledge base itself. INLEN also provides functions that facilitate manipulation of both the data and the knowledge base.

This research aims at developing a laboratory for studying the extraction of knowledge from large databases, and at developing a useful tool that can be applied to various practical discovery problems. INLEN is intended to demonstrate a system that is capable of examining large quantities of data, detecting trends, correlations and anomalies in the data, analyzing the importance of these discoveries, 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 AI and machine learning 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 database (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 database analysts in diverse fields.

The first stage of INLEN's implementation has already been completed, expanding upon the foundations of the QUIN, ADVISE and AURORA systems. In addition, many of the modules that will be adapted for use as operators in INLEN have been implemented as stand-alone systems or as parts of larger units. Other tools and the general integrated interface are under development. Future work will involve bringing these systems together and completing the control system to facilitate access to them in the form of simple, uniform commands. Research issues to be addressed include the performance of this methodology on larger databases and the optimization of the utility of the discovered knowledge.

Acknowledgments

The authors thank Michael Hieb, Isaac Huang, Gheorghe Tecuci and Brad U for their comments and criticism of the earlier drafts of this paper. This resear was done in the Artificial Intelligence Center of George Mason University. Tl activities of the Center are supported in part by the Defense Advanced Resear Projects Agency under the grants administered by the Office of Naval Researc No. N00014-87-K-0874 and No. N00014-91-J-1854, in part by the Office Naval Research under grants No. N00014-88-K-0397, No. N00014-88-K-022 No. N00014-90-J-4059, and No. N00014-91-J-1351, and in part by the Nation Science Foundation under grant No. IRI-9020266.

References

[1] Baim, P.W., "The PROMISE Method for Selecting Most Relevant Attributes for Inductive Learni Systems," Report No. UIUCDCS-F-82-898, Department of Computer Science, University of Illino Urbana IL, 1982.

[2] Baskin, A.B. and Michalski, R.S., An Integrated Approach to the Construction of Knowledg Based Systems: Experiences with ADVISE and Related Programs. In Guida, G. and Tasso, (Eds.), *Topics in Expert System Design*. Elsevier, Amsterdam, pp. 111–143, 1989.

[3] Bloedorn, E. and Michalski, R.S., "Data-Driven Constructive Induction in AQ17-DCI: A Methc and Experiments," *Reports of Machine Learning and Inference Laboratory*, MLI 89–12, Center f Artificial Intelligence, George Mason University, Fairfax, VA, 1992, to appear.

[4] Boose, J.H., and Gaines, B.R. (Eds.), *Proceedings of the 4th Knowledge Acquisition for Knowledg Based Systems Workshop*, Banff, Canada, 1989.

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

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

[7] Cestnik, B., Kokonenko, I., and Bratko, I., "ASSISTANT 86: A knowledge elicitation tool f sophisticates users," in *Proc. Second European Working Session on Learning*, Bratko, I. and Lavra N. (Eds.), Bled, Yugoslavia, 1987.

[8] Collins, A. and Michalski, R.S., The Logic of Plausible Reasoning: A Core Theory," *Cognitiv Science* vol. 13, pp. 1–49, 1989.

[9] Cramm, S.A., "ESEL/2: A Program for Selecting the Most Representative Training Events for Ir ductive Learning," Report NO. UIUCDCS-F-83-901, Department of Computer Science, Universit of Illinois, Urbana, IL, 1983.

[10] 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.

[11] Dietterich, T. and Michalski, R.S., Learning to Predict Sequences. In Michalski, R.S., Carbonel J.G. and Mitchell, T. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol. II. Morga Kaufmann Publishers, Los Altos, CA, pp. 63–106, 1986.

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

[13] Falkenhainer, B. and Michalski, R.S., Integrating Quantitative and Qualitative Discovery in th ABACUS System. In Kodratoff and Michalski, R.S. (Eds.), *Machine Learning: An Artificial Intelli gence Approach*, Vol. III. Morgan Kaufmann, San Mateo, CA, 1990.

[14] Greene, G., "Quantitative Discovery: Using Depedencies to Discover Non-Linear Terms," Master's Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1988.
[15] 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.
[16] International Intelligent Systems, Inc., *User's Guide to AURORA 2.0: A Discovery System*, International Intelligent Systems, Inc., Fairfax, VA, 1988.
[17] 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, 1987.
[18] 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 Databases, Detroit, MI, 1989.
[19] 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.
[20] 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.
[21] Kerschberg, L. (Ed.), *Expert Database Systems: Proceedings from the First International Workshop*, Benjamin/Cummings, Menlo Park, CA, 1986.
[22] Kerschberg, L. (Ed.), *Expert Database Systems: Proceedings from the First International Conference*, Benjamin/Cummings, Menlo Park, CA, 1987.
[23] Kerschberg, L. (Ed.), *Expert Database Systems: Proceedings from the Second International Conference*, Benjamin/Cummings, Menlo Park, CA, 1988.
[24] Kokar, M.M., Coper: A Methodology for Learning Invariant Functional Descriptions. In Michalski, R.S., Mitchell, T.M., and Carbonell, J.G. (Eds.), *Machine Learning*, Kluwer, Boston, 1986.
[25] Laird, J.E. (Ed.), *Proceedings of the Fifth International Conference on Machine Learning*, University of Michigan, Ann Arbor, MI, 1988.
[26] Langely, P., Bradshaw G.L., and Simon, H.A., Rediscovering Chemistry with the BACON System. In Michalski, R.S., Carbonell, J.G. and Mitchell. T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann, San Mateo, CA, 1983.
[27] 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.
[28] 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.
[29] Michalski, R.S., Theory and Methodology of Inductive Learning. In Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann, San, Mateo, CA, 1983.
[30] Michalski, R.S., "Toward a Unified Theory of Learning: Multistrategy Task-adaptive Learning," *Reports of Machine Learning and Inference Laboratory*, MLI 90-1, Center for Artificial Intelligence, George Mason University, Fairfax, VA, 1990.
[31] Michalski R.S. and Baskin, A.B., "Integrating multiple knowledge representations and learning capabilities in an expert system: The ADVISE system," in *Proc. 8th Int. Joint Conf. Artif. Intell.* Karlsruhe, West Germany, pp. 256–258, 1983.
[32] Michalski, R.S., Baskin, A.B., and Spackman, K.A., "A logic-based approach to conceptual database analysis," in *Sixth Ann. Sympo. Computer Appl. Medical Care*, George Washington University Medical Center, Washington, DC, pp. 792–796, 1982.

[33] 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.

[34] 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, 1985.

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

[36] 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 Comptuer Science, University of Illinois, Urbana, IL, 1978.

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

[38] 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.

[39] 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.

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

[41] Quinlan, J.R., Probabilistic Decision Trees. In Kodratoff, Y. and Michalski, R.S. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol. III. Morgan Kaufmann, San Mateo, CA, 1990.

[42] 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.

[43] Segre, A.M. (Ed.), *Proceedings of the Sixth International Workshop on Machine Learning*, Cornell University, Ithaca, NY, 1989.

[44] 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.

[45] 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.

[46] Stepp, R.E., "A Description and User's Guide for CLUSTER/2, a Programe for Conceptual Clustering," Department of Computer Science, University of Illinois, Urbana, IL, 1983.

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

[48] 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.

[49] Wnek, J. and Michalski, R.S., "An experimental comparison of symbolic and subsymbolic learning paradigms: phase I – learning logic-style concpets," in *Proc. First International Workshop Multistrategy Learning*, Harpers Ferry, WV, pp. 324–339, 1991.

[50] Wnek, J., Sarma, J., Wahab, A., and Michalski, R.S., Comparing Learning Paradigms via Diagrammatic Visualizatioin. In Ras, Z., Zemankova, M., and Emrich, M. (Eds.), *Methodologies for Intelligent Systems 5*, Elsevier, New York, pp. 428-437, 1990.

[51] Zhang, J., Integrating Symbolic and Subsymbolic Approaches: Learning Flexible Concepts. In Michalski, R.S. and Tecuci, G. (Eds.), *Machine Learning: A Multistrategy Approach*, Vol. IV. Morgan Kaufmann, San Mateo, CA, to appear.

[52] Zhang, J. and Michalski, R.S., "Combining Symbolic and Subsymbolic Representations in Learning Flexible Concepts: The FCLS System," *Reports of Machine Learning and Inference Laboratory*, Center for Artificial Intelligence, George Mason University, Fairfax, VA, to appear.

[53] Zytkow, J.M., "Combining many searches in the FAHRENHEIT discovery system," in *Proc. Fourth Int. Workshop Machine Learning*, Irvine, CA, pp. 281-287, 1987.