KNOWLEDGE EXTRACTION FROM DATABASES: DESIGN PRINCIPLES OF THE INLEN SYSTEM

by

K. Kaufman
R. S. Michalski
L. Kerschberg

Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems, ISMIS'91, Oct 16-19, 1991.

91-24

Presented at Sixth International Symposium on Methodologies for Intelligent Systems, ISMIS'91

KNOWLEDGE EXTRACTION FROM DATABASES: DESIGN PRINCIPLES OF THE INLEN SYSTEM

Kenneth A. Kaufman, Ryszard S. Michalski and Larry Kerschberg

Artificial Intelligence Center
George Mason University
Fairfax VA 22030

KNOWLEDGE EXTRACTION FROM DATABASES: DESIGN PRINCIPLES OF THE INLEN SYSTEM

Kenneth A. Kaufman, Ryszard S. Michalski and Larry Kerschberg

Artificial Intelligence Center
George Mason University
Fairfax VA 22030

ABSTRACT

The architecture of a large-scale system, INLEN, is presented as a methodology for the discovery of knowledge from facts. INLEN combines database, knowledge base, and machine learning methods within a uniform user-oriented framework. Data and various forms of knowledge are managed in a uniform way by using various operators. The system's knowledge is stored as *knowledge segments,* which are structures that link relational tables with rules, equations and/or hierarchies. A variety of machine learning programs are incorporated into the system to serve as high-level *knowledge generation operators.* These operators are used for generating various kinds of knowledge about the properties and regularities existing in the data. For example, such operators can hypothesize general rules from facts, 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, and determine a variety of statistical characteristics of the data. The management and discovery operators may be combined into macros or programs for repeated applications or automatic branching.

1. Introduction

In response to the rapid expansion and widespread use of database technology, there is a growing interest in developing new techniques for extracting knowledge from data. In particular, there is interest in developing methods that can go beyond the traditional statistical analyses, and produce symbolic data descriptions in addition to numerical ones. Such methods should be able to discover "conceptual" patterns in the data, suggest explanations for them, and generate plausible predictions.

This paper presents the ongoing research on the development of a system, INLEN, that employs a variety of machine learning and discovery techniques to extract useful knowledge from databases. In particular, the paper discusses ideas on how to integrate these techniques into a large-scale data analysis system, and presents the high-level architecture of the system. INLEN is built around a relational database, coupled with a knowledge base that contains facts, regularities, constraints and

hypotheses about the information stored in the database and the knowledge base itself. A piece of knowledge is stored in the form of a *knowledge segment*, a structure that combines data and knowledge of various types (e.g., rules, tables, equations, hierarchies, etc.)

INLEN is controlled by three sets of operators. The *data management operators* and *knowledge management operators* allow the user to control manually the contents of the data and knowledge bases respectively. The *knowledge generation operators* invoke various machine learning and discovery programs in order to generate new knowledge about the problem domain. These operators may be combined into larger constructs, macros or programs, in order to facilitate repeated knowledge discovery operations.

A knowledge generation operator, given a knowledge segment derives a new knowledge segment. Such a derivation is accomplished by various forms of inference, such as deduction, induction or analogy. In the INLEN design, the knowledge generation operators include operators 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, etc. These operators can be invoked by a user at any step of data analysis.

The development of a system such as INLEN is a complex task that consists of building a software environment for the incorporation and support of the functions of individual modules, and the development of these modules. Each module by itself is a complex program, which may represent an advanced machine inference or learning capability or a group of related capabilities. Consequently, the implementation of INLEN has to proceed incrementally.

This paper presents the ongoing development of INLEN. It emphasizes the system's architecture and the ways by which knowledge may be discovered using a system with such a design. A general description of the operators used by INLEN is presented, and the integration of these operators is discussed.

2. The Conceptual Structure of INLEN

The motivating idea behind the INLEN system is to provide a user with an integrated set of tools for manipulating both data and knowledge, and for extracting new or better knowledge from that

data and knowledge. To this end, INLEN integrates database, knowledge base and machine learning technologies. By integrating such components, INLEN can perform a wide range of functions. Current machine discovery tools typically have domain-independent, task-specific discovery capability. That is, a small amount of background knowledge allows a system to discover useful facts in diverse sets of data, but it is likely to be limited to the discovery of certain types of facts.

For example, a program for learning rules from examples, such as AQ [Michalski and Larson, 1983, eg.], can detect general regularities in a set of data. Specifically, it can discover patterns such as: "Most of the faulty components in this vehicle were made by producer X or producer Y during the period T." On the other hand, a program for conceptual clustering might formulate a classification such as: "Components in this vehicle can be divided into large items of type A or small items of types B and C." However, the rule discovery program will not be able to create a grouping from a set of objects, and conversely, the clustering program will be useless for differentiating between existing groups of objects.

The development of INLEN is based on the philosophy that task-specific discovery tools have evolved to such a point that they can be combined to create a more powerful discovery system. INLEN is a methodology for multitask discovery, using an architecture that can integrate diverse tools. By invoking different operators, a user may specify different discovery tasks to be performed.

INLEN thus can be viewed as an "intelligent data analysis assistant," that performs sophisticated data analysis operations, either under the directed guidance of a user, or partially autonomously. Specifically, such an assistant can conduct "conceptual" data analysis, search for unknown regularities, and propose explanations for the patterns discovered. To reflect the needs of the user, the system's knowledge base can be equipped with criteria that characterize the classes of patterns that are important to a specific user.

The approach that we are employing 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. Data and knowledge management, 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 can exhibit a synergistic effect in extracting useful knowledge from data, and have an increased potential for making discoveries. A machine learning system may also be potentially useful in formulating explicit criteria that experts are using implicitly in evaluating the "interestingness" of a pattern.

3. Data and Knowledge Bases

INLEN is designed under the assumption that the data is stored in a relational database system, and that some locations in a table may be marked "?" to indicate unknown values, or marked "N/A" denoting the non-applicability of some attributes to some objects. It is also assumed that encoded in the system is rudimentary background knowledge about the variables that characterize the data, the values they can have, and the relationships among those values (linear, hierarchical, etc.), and possibly among the variables. Given such a relational table, the INLEN operators will try to discover useful facts about the objects represented in the table.

In order to maintain the database itself, a set of data management operators (DMOs) are available in INLEN. These operators perform traditional relational operations, such as PROJECT, SELECT, DELETE, CHANGE, JOIN and UNION. The full set of data management operators is depicted in Figure 1.

The underlying knowledge representation in INLEN is *knowledge segments*, which are flexible structures for storing background or discovered knowledge about the facts in the database. Knowledge segments link relational tables with rules, equations and/or hierarchies. 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, understanding and manipulating such structures.

A user may alter the knowledge base directly by invoking one of a set of knowledge management operators (KMOs, also depicted in Figure 1). These operators are designed to be similar to the data management operators at the user level, while concealing internally the instructions specific to the manipulation of the knowledge base.

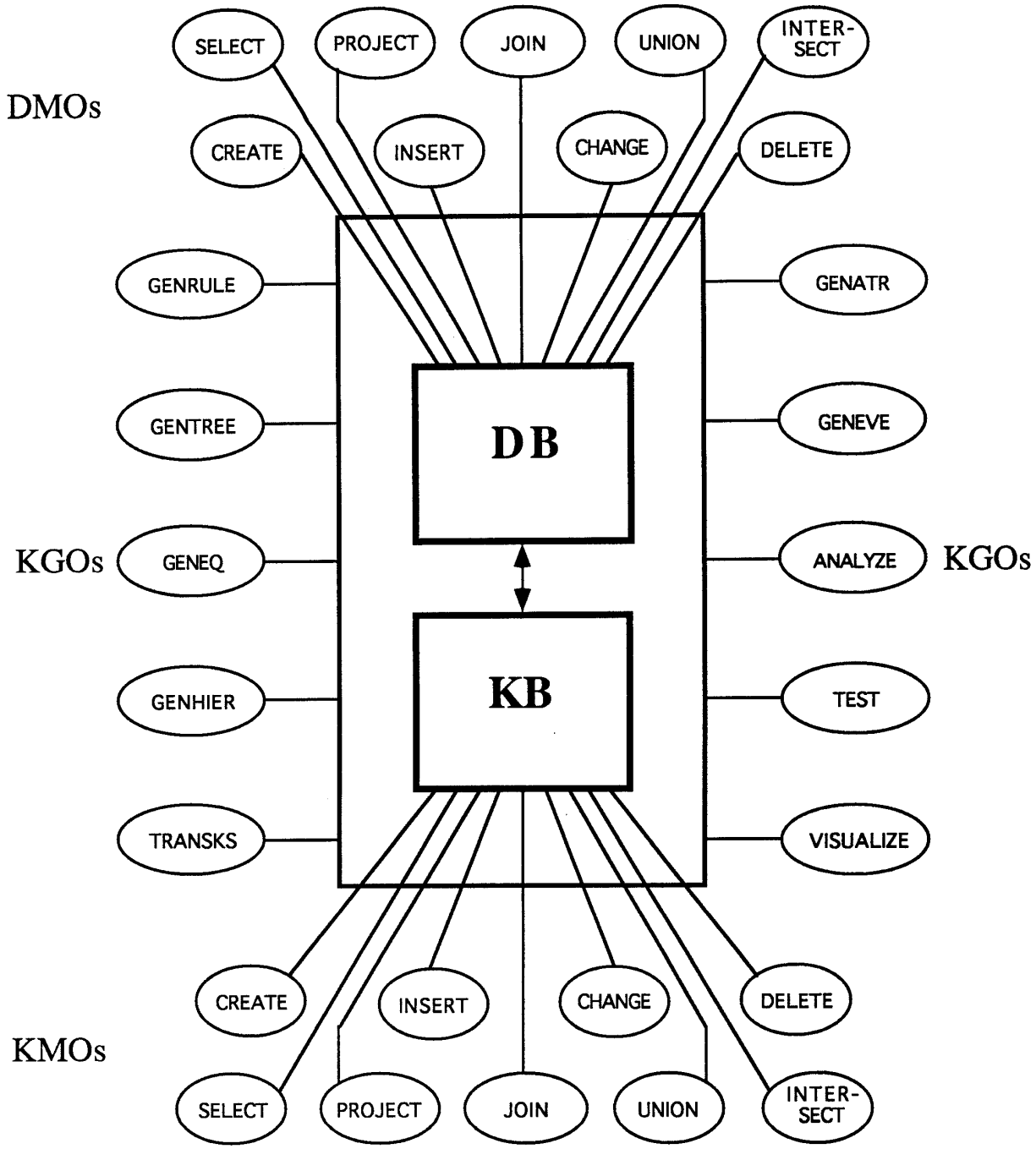

Figure 1 INLEN's Top-Level Functional Architecture

4. Knowledge Generation Operators

INLEN's knowledge generation operators (KGOs) perform inferences on knowledge segments in order to create new knowledge. As part of their function, these operators 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 knowledge segments 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 descriptions of the individual operators; they may be assumed to be part of any KGO.

In general, a knowledge generation operator takes one table from the database and one or more knowledge segment 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.

The KGOs It is practical to group these operators conceptually into ten high-level groups based on the types of output they generate (see Figure 1). Those whose primary output consists of rules are separated from those that generate new tuples for a relational table, 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; examples of programs that perform these particular inference tasks are cited in [Kaufman, Michalski and Kerschberg, 1990]. A description of the KGO classes and the individual operators follows:

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. There are four different operators in the GENRULE class.

CHARSET (Characterize Set) determines a description characterizing a class of entities by discovering characteristic rules. DIFFSET (Differentiate Set) takes one set of objects (each object represented as a tuple in a relational table of data or metaknowledge) as a primary input, and one or more sets of objects as a controlling input, and induces general rules that encapsulate the differences between the primary set and the other sets.

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. Similarly, DIFFSEQ (Differentiate Sequence) discovers differences between two or more sequences of objects or events. Given input consisting of a primary sequence and one or more other sequences, this operator will seek rules that encapsulate the differences between the primary sequence and the other input sequences.

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 examples, while RULETREE (Rule to Tree) organizes a set of decision rules into a tree.

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 the rules that determine 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.

TRANSKS: Transform Knowledge Segments

The TRANSKS 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. ABSTRACT modifies its input knowledge segment by removing details from its description, and CONCRETIZE specifies details of an abstract concept description, while GENERALIZE and SPECIALIZE affect the set size covered by the input knowledge segment. Examples of these four operators are shown by Kaufman, Michalski and Kerschberg [1990]. IMPROVE detects exceptions to existing knowledge in its input examples, and refines the knowledge segments accordingly.

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. CONATR (Construct Attribute) applies mathematical operators specified in its arguments in order to combine variables into useful composites.

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 in which a table's tuples belong. SELEVE (Select Event) determines the examples 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, consisting of the most promising examples from the input table. CONEVE (Construct Event)

searches the example set or event space for elements satisfying some selection criteria. PREDVAL (Predict Value) speculates on likely values for unknown attributes of incomplete or hypothetical data elements. SIMILIZE (Find Similar) seeks out events or relationships that are similar to the input in some defined sense.

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, and RELEVE (Relate Events) similarly determines relationships among elements in a relational table. RELKS (Relate Knowledge Segments) discovers relationships such as inclusion, disjointedness, correlation, generalization and abstraction within a set of knowledge segments. GENSTAT (Generate Statistics) uses existing programs to perform a statistical analysis of the data in order to determine its various statistical properties.

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. Consistency implies that no event in the example set is covered by two different rules.

VISUALIZE: Diagrammatic Visualization

VISUALIZE displays a set of data, as specified by rules or transformations of it, graphically on the screen. The output from this operator appears as a two-dimensional representation of the event space, with the input set highlighted. VISUALIZE uses the DIAV diagrammatic visualization methodology, currently being developed [Wnek et al, 1990; Wnek and Michalski, 1991].

5. Macrooperators and Data Analysis Programs

In order to facilitate the repeated execution of certain operator sequences by users, INLEN is designed to provide mechanisms for creating macrooperators and high-level data analysis

programs. Macrooperators 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. Among the research and design issues to be addressed during INLEN's prototype development and testing will be which sequences of operators are most useful for inclusion as standard macros in the INLEN package, and what is an effective scheme in which users can develop, write and store their own macrooperators.

It may also be the case that there is a repeatable application 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 data management operator 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.

As a more complex example, consider the task of determining regularities in a large group of patients, whose records have been stored in a large hospital database that contains very detailed information on the patients. Suppose a contagious disease that primarily affects male patients is discovered in the hospital, and the disease is only detected with certainty via an extensive and costly series of medical tests. We would like to use INLEN to quickly determine which patients can be excluded from having to undergo those tests, due to their clear lack of indicators of the disease. A knowledge generation operator such as DIFFSET can find simple rules that can differentiate between those who will test negative and those who may test positive. Other operators can vouch for the accuracy of the rules and apply them to untested patients. The hope in this case is that by employing INLEN, excessive testing, quarantining, and/or spread of the disease can be avoided. The algorithm for a possible data analysis program to perform such a task is illustrated in Figure 2. It should be reemphasized that the depicted algorithm is a simplification; the operator calls will generally include other optional parameters to guide the depth of their searches and their selection of a "best" output.

The initial step is to determine the number of variables in this particular data table, and if this number is likely to generate too lengthy a search, the SELATR operator is called upon to select

only the most promising variables in the table. These variables constitute the attributes in a reduced data table (or more precisely, a limited view of the original table.) Similarly, SELEVE may be invoked to create manageable subtables containing representative records of male patients who tested positive for the disease on the comprehensive medical examination and male patients who tested negative for it.

These examples are given as input to the DIFFSET operator, which hypothesizes rules to discriminate between the cases who have tested positive and those who have tested negative. In order to determine the accuracy of these rules, the TEST operator is invoked to check the newly created rules against all available male patients whose medical test results are known. The frequency of false negatives from this analysis can be then weighed against a threshold; if it is below the threshold, we conclude that INLEN has found an acceptable way to eliminate some patients from suspicion of having the disease.

```
If  #ATR(Patients)  >  20                                  ** If the set of attributes is too
    Then  Begin                                            **      large (> 20) ...
          SELATR(Patients,20,Inppat).                      ** Select the 20 most represen-
          End                                              **      tative attributes into Inppat
    Else  SELECT(Patients:  all,Inppat).                   ** Else there are few enough
                                                           **      variables
SELECT(Inppat:  [sex=male][tested=positive],  Pos).        ** Find all positive examples
SELECT(Inppat:  [sex=male][tested=negative],  Neg).        ** Find all negative examples
If  #EVE(Pos)  >  50                                       ** Are there >50 positive cases?
    Then  SELEVE(Pos,  50,  Postest)                       ** Get 50 typical positive cases
Else  SELECT(Pos:  all,Postest).                           ** Test all positive examples.
If  #EVE(Neg)  >  50                                       ** Are there >50 positive cases?
    Then  SELEVE(Neg,  50,  Negtest)                       ** Get 50 typical negative cases
Else  SELECT(Neg:  all,Negtest).                           ** Test all negative examples.
DIFFSET(Postest,  Negtest,  Rules).                        ** Determine discriminant rules
SELECT(Patients:  [sex=male][tested<>?],  TesPat).         ** Find all examples with
                                                           **      known test results
TEST(TesPat,  Rules,  accuracy).                           ** Apply rules to tested cases
If  (accuracy  >  Threshold)                               ** If the rules are good enough
    Then  Begin                                            ** Then ...
          SELECT  (Patients:  [sex=male][tested=?],  Newmen)
          PREDVAL(Newmen,  Rules,  Probable).              ** Extrapolate for other men and
          SELECT(Probable,  [tested=negative],  Output).
          End.                                             ** Report those who are safe
```

Figure 2. The algorithm for a data analysis program

In this case, INLEN can asked to infer test results for the male patients who have not been tested, using the rules now in its knowledge base. Since the accuracy of the rules in predicting negative cases was greater than an assumed threshold, the system lists the patients found negative according to INLEN's rules. They will not have to be tested.

6. The Implementation of INLEN

The purpose of this research is to develop an environment in which different knowledge discovery techniques can be coordinated into a larger, multi-purpose tool. This coordination is carried out through a uniform user interface, menu-driven parameter selection, and the restriction of task-specific code to levels hidden to the user. Under such a methodology, user- or data-driven calls to the individual operators can direct the discovery of useful facts without having to adjust strategies to the different operators.

INLEN is a complex large-scale system composed of many intricate modules, some of which can serve as powerful stand-alone systems. The design and implementation of the system builds upon the development of the QUIN system [Michalski, Baskin and Spackman, 1982; Spackman, 1983]. 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 some cases, this includes primarily a change of the program interface, while in others, the programs have to undergo major modifications or be redeveloped from scratch. Finally, some of the knowledge generation operators are still at the stage of research and initial implementation. Therefore, INLEN implementation is regarded as a multi-stage task.

The first stage of development (version INLEN-0) includes a knowledge base of simple decision rules, a relational database, and an extensive user-oriented menu-based graphical interface. The knowledge generation operators incorporated in this system include preliminary versions of such operators as: CHARSET, DIFFSET, IMPROVE, TEST and PREDVAL. The system has been implemented on an AT-compatible computer.

The second stage involves the development of a larger prototype on a Sun platform that integrates a full-fledged knowledge base with a commercial-grade database. This prototype is oriented toward a specific task, namely the discovery of interesting anomalies and relationships in an economic database. The system includes most of the knowledge generation operators described in Section 4, and a new system interface.

The third stage will involve the development and implementation of the remaining operators, allowing for a greater inferential capacity. This stage of development will also include any modifications to the structure of the system's components based on the results generated during the previous stage.

7. Conclusion

Most research on the discovery of knowledge in databases is concerned with some specific type of discovery or knowledge extraction. The main aim of this paper is to discuss design ideas and an architecture of a system that integrates many different machine learning and discovery programs. Specifically, the paper presents the architecture of INLEN, a large-scale system capable of performing a wide variety of inferential operations on data in order to discover interesting regularities in them. These regularities can be detected in qualitative data, quantitative data, and in the knowledge base itself. In addition, INLEN provides operators that facilitate the manipulation of both the data and the knowledge base, as well as macrooperators 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.

INLEN implements a number of novel ideas. It integrates diverse knowledge generation operators that permit a user to search for various kinds of relationships and regularities in the data. It thus can exploit the strengths of different learning and discovery programs, reduce its limitation to specific tasks, and attain the capability for multistrategy learning and discovery.

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.

The first stage of INLEN's implementation has already been completed, expanding upon the foundations of the QUIN, ADVISE [Michalski et al., 1987] and AURORA [INIS, 1988] systems. In addition, many of the modules to be incorporated 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.

ACKNOWLEDGEMENTS

The authors thank Michael Hieb, Gheorghe Tecuci and Brad Utz for their comments and criticism of the earlier versions of this paper.

This research was done in the Artificial Intelligence Center of George Mason University. The activities of the Center are supported in part by the Defense Advanced Research Projects Agency under grant, administered by the Office of Naval Research No. N00014-87-K-0874, in part by the Office of Naval Research under grant No. N00014-88-K-0226, and in part by the Office of Naval Research under grant No. N00014-88-K-0397.

References

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

Kaufman, K., Michalski, R.S. and Kerschberg, L., "An Architecture for Knowledge Discovery from Facts: Integrating Database, Knowledge Base and Machine Learning in INLEN," submitted to *Reports of Machine Learning and Inference Laboratory*, 1990.

Michalski, R.S., "Toward a Unified Theory of Learning: Multistrategy Task-adaptive Learning," MLI Report No. 90-1, Artificial Intelligence Center, George Mason University, 1990.

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, November 1-2, 1982, pp. 792-796.

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, Jan. 1987.

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, Jan. 1983.

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, 1983.

Wnek, J. and Michalski, R.S., "A System for Visualization of Learning and Inference Processes," in preparation.

Wnek, J., Sarma, J., Wahab, A. and Michalski, R.S., "Comparing Learning Paradigms via Diagrammatic Visualization," *Proceedings of International Symposium on Methodologies of Intelligent Systems ISMIS-90*, Ras, Z., Zemankova, M., Emrich, M. (Eds.), Elsevier Press, 1990.