THE INLEN SYSTEM FOR EXTRACTING KNOWLEDGE FROM DATABASES: GOALS AND GENERAL DESCRIPTION

by

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

Reports of the Machine Learning and Inference Laboratory, MLI 89-6, School of Information Technology and Engineering, George Mason University, Fairfax, VA, 1989.

The INLEN System for Extracting Knowledge from Databases: Goals and General Description

K. Kaufman, R. S. Michalski,
J. Zytkow and L. Kerschberg

MLI 89-10

For submission to IJCAI-89 Workshop on Knowledge Discovery in Databases, August 20, 1989.

THE INLEN SYSTEM FOR EXTRACTING KNOWLEDGE FROM DATABASES: Goals and General Description

Kenneth A. Kaufman, Ryszard S. Michalski, Jan M. Zytkow, Larry Kerschberg

Artificial Intelligence Center
George Mason University
Fairfax VA 22030

ABSTRACT

The INLEN system combines database, knowledge base, and machine learning technologies to provide a user with an integrated system of tools for conceptually analyzing data and searching for interesting relationships and regularities in them. Machine learning techniques are used for determining general descriptions from facts, creating conceptual classifications of data, selecting the most relevant attributes, determining the most representative examples, and discovering equations binding numeric variables, as well as formulating applicability conditions for these equations.

1. Introduction

This paper briefly describes the goals and general design of the system INLEN (**In**ference and **Le**arning) for conceptually analyzing databases and discovering interesting relations and patterns in them. The system represents an extension and further development of earlier systems such as QUIN (Query and Inference), which was designed as a database interface incorporating a number of inductive learning techniques [Michalski and Baskin, 1983; AGASSISTANT [Katz, Fermanian and Michalski, 1986], a shell for developing agricultural expert systems; and AURORA [INIS, 1988], a general-purpose PC-based expert system shell with learning and discovery capabilities, designed by Michalski and Katz.

INLEN integrates a relational database, a rule base, and a number of machine learning capabilites. The latter ones enable the system to create conceptual descriptions of facts in the database, propose various classifications of data, discoveri rules and unknown regularities, formulate equations together with the conditions of their applicability, as some other. We present here a general system design and explain all basic functions.

2. INLEN System Design

INLEN combines expert database technology and machine learning systems in order to create an environment in which different types of data analysis can be performed. It integrates several advanced machine learning capabilities which until now have existed only as separate experimental programs. Many learning systems are capable of but a narrow subset of what can be learned from factual data. By integrating a variety of these tools, a user will have access to a powerful and versatile system.

INLEN evolved from the QUIN system [Michalski and Baskin, 1983; Spackman, 1982], a combined database management and data analysis environment, which was designed both as a stand-alone system and as a tool for incorporation into the ADVISE meta-expert system [Michalski et al, 1987]. In the last few years, new tools have been developed and made available for INLEN's use, both in inductive learning and in expert database management.

The general design of INLEN is shown in Figure 1. The INLEN system consists of a relational database for storing known facts about a domain, and a knowledge base for storing constraints, production rules, hierarchies, equations, behavioral knowledge, and triggers for functions which are to be activated by certain conditions in the data or knowledge base. The knowledge base can contain not only knowledge about the contents of the database, but also metaknowledge for the dynamic upkeep of the knowledge base itself.

INLEN provides a user with three sets of functions (operators): *data management operators*, *knowledge management operators*, and *knowledge generation operators*. The data management operators (DMO) are standard relational operators for accessing, retrieving and manually altering the information in a relational database. The knowledge management operators (KMO) perform similar tasks on the rules in the knowledge base for situations in which manual input, access or adjustments are desired. The knowledge generating operators (KGO) interact with both the database and the knowledge base; they are a collection of machine learning systems used to generate or improve the system's knowledge base and the metaknowledge necessary for efficient operation. Examples of some of the knowledge generating operators will be shown in the following section.

A brief description of each of INLEN's DMO, KMO and KGO functions follows.

DMO Functions

The DMO functions form a standard set of operations for the purpose of manipulating the system's collection of known facts. They are listed here for the sake of completeness.

CREATE will generate a new relational table. It takes an attribute list as an argument.

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

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

DELETE removes rows or columns from a table, as specified respectively by SELECT-R or SELECT-C operations. Alternatively, entire tables may be removed from the system.

SELECT-C selects a number of columns from a relational table, and returns those columns in the order desired. This is the equivalent to PROJECT operations in a standard relational system.

SELECT-R performs in the manner of a relational SELECT operation. It chooses a subset of the rows of a relational table based on some selection criteria.

JOIN, in its simplest form, creates the Cartesian product of two tables. A following SELECT-R operation may then be embedded in the join. For example, the natural join of two tables which have a column (or more) in common is the subset of the set of rows of the

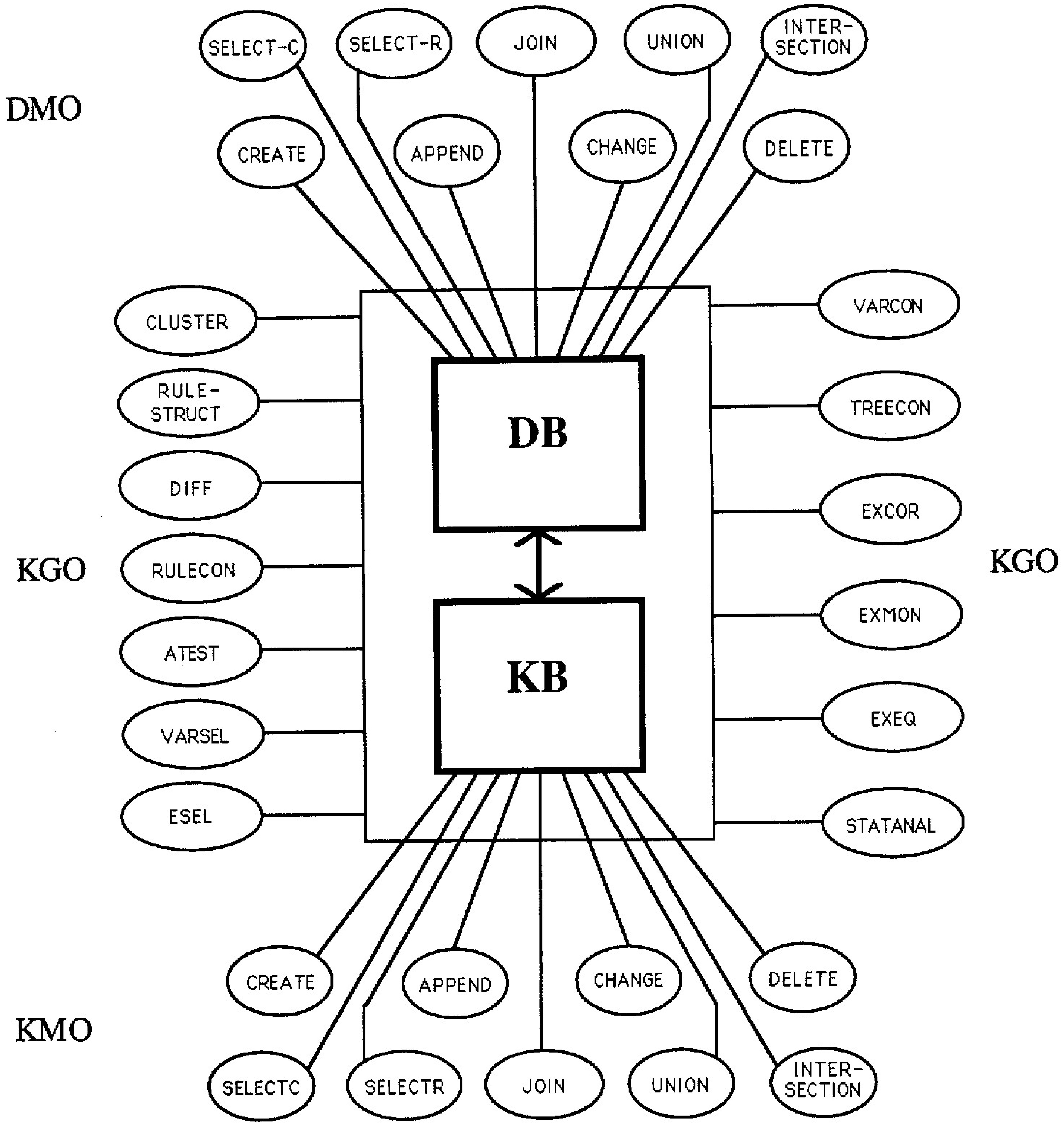

Figure 1. A Functional Diagram of INLEN

join in which the matching columns have the same value. When applied to an INLEN database, JOIN will produce the join of two tables.

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.

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

KMO Functions

These functions allow the INLEN knowledge base to be handled in the manner of any other data set. A set of production rules can often be viewed a table; each rule is represented by a row, and each attribute is represented as a column. The (i,j)th entry in the table would then be the legal values of the ith attribute in the jth rule. Knowledge management operators permit manual adjustments to the rule base, as well as queries for selective access.

CREATE will generate a new rule set. The rule set will be empty until rules are added using either an APPEND operation or one of the KGO techniques.

APPEND may be used for the manual addition of a new rule to a rule set..

CHANGE is used for the manual alteration of some of the conditions of one or more rules in the knowledge base.

DELETE may remove entire rules (for a "row delete") or all references to certain attributes (in the case of a "column delete") from a rule base. Alternatively, an entire rule base may be erased by giving no qualifying conditions.

SELECT-C selects a number of columns from a knowledge base. If the rule base is thought of as being in the form of a table with rules forming the rows and values of variables in its conditions and actions forming the columns, this operation can be thought of as generating partial rules consisting only of the selected variables.

SELECT-R generates a list of the rules which satisfy the given criteria.

JOIN, applied to two rule sets in a knowledge base, creates the Cartesian product of these rule sets. For example, the natural join of two rule sets will consist of pairs of rules from the rule sets which have a selected condition or action in common.

UNION will generate a list of all the rules present at least once in each of the rule sets supplied.

INTERSECTION generates a list of the rules common to both parameter rule sets.

KGO Functions

The KGO operations allow for inferences to be made on the data in the database and the rules already present in the knowledge base. It should be noted that these may be applied in different manners depending on the nature of their inputs and outputs (whether they are decision tables, rules, or both.) In general, these operations may be thought of as mappings from the database and/or the knowledge base into the (new) database and/or knowledge base. The scope and use of these functions are described below:

CLUSTER performs conceptual clustering in order to create logical groupings among a collection of objects or events. The CLUSTER operation divides a group of examples from the database into two or more subgroups, and returns this grouping and the system's rationale for it. User-defined parameters may influence the selection of the groupings. Detailed descriptions of the clustering algorithm may be found in [Michalski, Stepp and Diday, 1981; Stepp, 1983 and Stepp, 1984].

RULESTRUCT creates groupings within a set of decision rules. For example, rules diagnosing cancer of the liver and cirrhosis of the liver might be grouped together as rules describing liver ailments. RULESTRUCT is another form of the clustering algorithm, this time applied to the knowledge base, rather than to example data.

DIFF takes several groups of objects and attempts to find rules which define why the objects belong to the groups they are in. The AQ system, which is capable of performing this function, is described in [Michalski & Larson, 1983]. The rules produced will be discriminant, ie, sufficient to explain what characteristics one class of objects contains that are not present in others.

RULECON discovers descriptions which characterize a set of input examples. This function also falls into the domain covered by AQ and similar systems. Here, the emphasis is on finding specific characteristics present in all examples of a class of object, without concern over whether they might be present in other classes.

ATEST [Reinke, 1984] tests a set of decision rules for consistency and completeness. Consistency implies that no event in the example space is covered by two different rules. Completeness refers to the condition that every possible example will be covered by the conditions applying to at least one rule. In many cases, consistency and/or completeness in some or all of the knowledge base will be desirable.

VARSEL [Baim, 1982] attempts to find attributes most likely to be important in differentiating between several classes of events. By pruning the example descriptions in such a manner, this can reduce the computation time required by the CLUSTER or DIFF operator.

ESEL, like VARSEL, prunes an example set, but it operates on entire examples, rather than their attributes. Promising examples are returned as output, while others are rejected. [Cramm, 1983] describes ESEL in detail.

VARCON [Davis, 1979] attempts to use mathematical operators to combine variables into useful composites. For example, there may be a case in which the sum of two variables is more useful than either individual value.

TREECON [Michalski, 1978; Layman, 1979] takes a set of rules or decision examples, and organizes them into a decision tree, which may be a more efficient way to store the knowledge.

EXCOR discovers correlations between the values of attributes in a set of examples. It is implemented using the AQ series of programs.

EXMON seeks out monotonic relations between attributes in a set of examples, and in doing so, may discover an interesting relationship within the data.

EXEQ discovers equalities that describe numeric data in a set of examples. ABACUS [Falkenhainer and Michalski, 1986] is one example of a system that employs such a methodology in order to discover relationships that hold among quantitative values.

STATANAL performs a statistical analysis of the data in order to find correlations and simple causal effects.

3. An Illustration of Selected KGO Functions: CLUSTER, DIFF, EXEQ

We now demonstrate the capabilities of some the CLUSTER, DIFF and EXEQ operations in order to present a representative example of KGO abilities.

CLUSTER

CLUSTER is capable of creating groupings of objects or events, and when used recursively, can generate an entire taxonomy. Unlike traditional clustering methods, CLUSTER also returns the rules which define its grouping. One example of conceptual clustering is shown in [Michalski & Stepp, 1983], involving the classification of microcomputers. Variables considered include the type of processor, the amount of RAM, the ROM size, the type of display, and the number of keys on the keyboard. Dividing the examples into two groups, the system grouped them according to RAM size and keyboard, while clustering into three groups was based on the other three variables. Table 1 demonstrates this example of CLUSTER's performance. In both experiments, the input to CLUSTER was a table of the characteristics of the microcomputers, and the output consisted of the groupings, plus rules and cross-reference tables to be added to the knowledge base.

DIFF

The DIFF operator is based on the AQ learning method which has been effectively used for many rule learning tasks in areas such as medicine, agriculture, physics, computer vision,chess, etc. One recent application to diagnose potential breast cancers given a few training examples is described in [Michalski, Mozetic et al, 1986]. The rules generated were found to satisfactorily encapsulate expert knowledge in the domain. A DIFF application to concisely describe the groups created by the clustering algorithm in Table 1 is demonstrated in Table 2. The groups, without any explanation are given as input, and DIFF creates rules which discriminate between these groups. Note that while not all of the rules are identical to what CLUSTER found, they are equally valid, and that a redundant condition (the processor type of the third group of the 3-grouping) is not included.

While this example showed an application of DIFF to create the *discriminant* rules for groups of examples, DIFF may also be used to determine *characteristic* rules that describe classes of events [Michalski, 1983]. It will then return optimized rules, which will highlight the differences between the classes. In larger example sets, there may be large differences between characteristic and discriminant rules.

EXEQ

Two numerical discovery systems are incorporated into the EXEQ modules, ABACUS and FAHRENHEIT. The first version of ABACUS, described in [Falkenhainer & Michalski, 1986],

INPUT						OUTPUT	
Microcomputer	Display	RAM	ROM	Processor	No_Keys	2-Group	3-Group
Apple II	Color_TV	48K	10K	6502	52	1	1
Atari 800	Color_TV	48K	10K	6502	57-63	1	1
Comm. VIC 20	Color_TV	32K	11-16K	6502A	64-73	1	2
Exidi Sorceror	B/W_TV	48K	4K	Z80	57-63	1	2
Zenith 118	Built_in	64K	1K	8080A	64-73	2	3
Zenith 1189	Built_in	64K	8K	Z80	64-73	2	3
HP 85	Built_in	32K	80K	HP	92	1	2
Horizon	Terminal	64K	8K	Z80	57-63	1	2
Challenger	B/W_ TV	32K	10K	6502	53-56	1	1
O-S 11 Series	B/W_TV	48K	10K	6502C	53-56	1	2
TRS-80 I	B/W_TV	48K	12K	Z80	53-56	1	1
TRS-80 III	Built_in	48K	14K	Z80	64-73	1	1

CLUSTER operator takes as the input the relational table, marked INPUT, and a parameter requiring it to partition the rows in the table into 2- and then into 3- group clusterings. The two rightmost columns show the partitions generated. The CLUSTER also generates rules describing the groups, stored in the KB:

2-Group clustering:
[Group 1] <== [RAM = 16K..48K] or [No_Keys ≤ 63]
[Group 2] <== [RAM = 64K] & [No_Keys > 63]

3-Group clustering:

[Group 1] <== [Processor = 6502 v 8080A v Z80] & [ROM = 10K..14K]
[Group 2] <== [Processor = 6502A v 6502C v HP] or [ROM = 1K..8K] & [Display ≠ Built_in]
[Group 3] <== [Processor = 6502 v 8080A v Z80] & [ROM = 1K..8K] & [Display = Built-in]

Table 1. Example of a CLUSTER Operation

Microcomputer	Display	RAM	ROM	Processor	No_Keys	2-Group	3-Group
Apple II	Color_TV	48K	10K	6502	52	1	1
Atari 800	Color_TV	48K	10K	6502	57-63	1	1
Comm. VIC 20	Color_TV	32K	11-16K	6502A	64-73	1	2
Exidi Sorceror	B/W_TV	48K	4K	Z80	57-63	1	2
Zenith 118	Built_in	64K	1K	8080A	64-73	2	3
Zenith 1189	Built_in	64K	8K	Z80	64-73	2	3
HP 85	Built_in	32K	80K	HP	92	1	2
Horizon	Terminal	64K	8K	Z80	57-63	1	2
Challenger	B/W_ TV	32K	10K	6502	53-56	1	1
O-S 11 Series	B/W_TV	48K	10K	6502C	53-56	1	2
TRS-80 I	B/W_TV	48K	12K	Z80	53-56	1	1
TRS-80 III	Built_in	48K	14K	Z80	64-73	1	1

DIFF takes as input a relational table in which the last column indicates group (class) membership. The DIFF operator tries to rediscover the rules, invented by CLUSTER, from the examples of groups:

Rules for 2-Group differentiation:

[Group 1] <== [Display ≠ Built_in] or [ROM ≥ 14K]
[Group 2] <== [RAM = 64K] & [No_Keys = 64-73]

Rules for 3-Group differentiation:

[Group 1] <== [Processor = Z80 v 6502] & [ROM = 10K..14K]
[Group 2] <== [Processor = 6502C v 6502A v HP] or [ROM = 4K..8K] & [Display = B/W_TV v Termin
[Group 3] <== [ROM = 1K..8K] & [Display = Built-in]

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

Table 2. Example of a DIFF Operation

is capable of learning equations which fit a set of tabular data. It is also capable of subdividing a set of examples into partitions in which different rules apply, and of coping with noisy data. The recent version, ABACUS-2 [Greene, 1988], has expanded the capabilities of ABACUS; non-linear regularities may now be discovered.

ABACUS has discovered equations to describe such things as planetary motion, the distances between atoms in a molecule, and Stoke's Law of falling bodies, which computes the velocity of an object falling through different media. Stoke's Law is demonstrated in Table 3. As Table 3 indicates, the velocity of an object falling through a fluid is governed by an equation much different from the equation which describes its velocity falling through a vacuum. ABACUS will be able to find the equations from both classes.

Another system planned for incorporation into the EXEQ module of regularity inference is FAHRENHEIT [Koehn & Zytkow, 1986; Zytkow, 1987]. It conducts its own experiments in the effort to discover regularities in a data set. While experimentation can be replaced by database queries, this strategy does not promise success because the data within databases are sparse and most queries will produce no answer. The problem can be bypassed by the use of an additional module that selects data from the database and organizes them into a tree structure. FAHRENHEIT does not really care about making experiments. It only cares that enough data were collected so that its partitioner and curve-fitter receive enough datapoints.

FAHRENHEIT is capable of performing several tasks. First, it can find a multidimensional numerical regularity (law). If the sequence of data cannot be summarized by a single regularity, the system can partition the data and find a number of regularities in result. Second, FAHRENHEIT finds the scope of any regularity it discovers. The scope of a law is described by boundaries that separates situations that obey the law from the situations that do not. Third, FAHRENHEIT is able to find an area in which no regularity has been detected yet, and conduct a search for regularity in such area. It continues in that manner until the whole space spanned by the independent variables is covered with regularities. All these capabilities are accompanied by data structures for the storage of detected regularities. The system can be used as a whole, or in separate parts in order to perform smaller tasks. This may be especially useful if the human operator wishes to develop insights into a system's operation or wishes to approve results before going on to the next step.

4. Conclusion

INLEN provides an integrated system capable of performing a wide variety of inductive learning techniques in order to analyze data in a relational database. Knowledge can be extracted from qualitative data, quantitative data, and the knowledge base itself. In addition, INLEN provides functions which facilitate manipulation of both the data and the knowledge base.

Many of INLEN's modules have already 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.

Substance	Radius (m)	Mass (kg)	Height (m)	Time (s)	Velocity (m/s)
Vacuum	0.05	1	6	0.1	0.98453
Vacuum	0.05	2	2	0.4	3.93812
Vacuum	0.10	1	3	0.5	2.95359
Vacuum	0.10	2	7	0.1	0.98453
Glycerol	0.05	1	5	0.1	19.112
Glycerol	0.05	2	8	0.3	38.224
Glycerol	0.10	1	6	0.5	9.556
Glycerol	0.10	2	7	0.2	19.112
CastorOil	0.05	1	9	0.4	14.672
CastorOil	0.05	2	3	0.1	29.344
CastorOil	0.10	1	5	0.3	7.336
CastorOil	0.10	2	8	0.5	14.672

ABACUS searches for relationships among the data objects. It discovers that equations for the ball's velocity exist, but they depend on the medium through which the ball is falling.

Here are the rules ABACUS discovered:

If [Substance = Vacuum] **then** $v = 9.8175 * t$

If [Substance = Glycerol] **then** $v * r = 0.9556 * m$

If [Substance = CastorOil] **then** $v * r = 0.7336 * m$

Table 3. ABACUS Discovers Stoke's Law

Acknowledgements

The authors would like to thank Peter Aiken, Kathleen Byrd, Joyce Ralston, Pawel Stefanski and Jianping Zhang for their assistance in this project.

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, N00014-85-K-0878, 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

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

R.L. Blum, "Automating the Study of Clinical Hypotheses on a Time-Oriented Data Base: The RX Project", Report No. STAN-CS-79-816, Department of Computer Science, Stanford University, Stanford CA, Nov. 1979.

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

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

B. Falkenhainer and R.S. Michalski, "Integrating Quantitative and Qualitative Discovery: The ABACUS System", Report No. UIUCDCS-F-86-967, Department of Computer Science, University of Illinois, Urbana IL, May 1986.

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

J. Hong, I. Mozetic and R.S. Michalski, "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, May 1986.

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

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

Koehn, B.W. & Zytkow, J.M. (1986) Experimenting and theorizing in theory formation. Proceedings of The ACM Sigart International Symposium on Methodologies for Intelligent Systems, Knoxville, 296-307.

Langley, P., Simon, H. A., Bradshaw, G. L. & Zytkow, J. M. (1987) Scientific discovery: Computational explorations of the creative processes. Cambridge, MA: MIT Press.

T.C. Layman, "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.

R.S. Michalski, "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, March 1978.

R.S. Michalski, "Theory and Methodology of Inductive Learning", in *Machine Learning: An Artificial Intelligence Approach*, Michalski, Mitchell, Carbonell eds, Morgan Kaufmann, 1983.

R. S. Michalski and A. B. Baskin, "Integrating Multiple Knowledge Representations and Learning Capabilities in an Expert System: The ADVISE System," Proceedings of the 8th IJCAI, Karlsruhe, West Germany, August 8-12, 1983, pp. 256-258.

R. S. Michalski, A. B. Baskin and K. A. Spackman, "A Logic-based Approach to Conceptual Database Analysis," 6th Annual Symposium on Computer Applications in Medical Care (SCAMC-6), George Washington University Medical Center, Washington, DC, November 1-2, 1982, pp. 792-796.

R.S. Michalski, A.B.Baskin, C. Uhrik and T. Channic, "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.

R.S. Michalski, L. Iwanska, K Chen, H. Ko and P. Haddawy, "Machine Learning and Inference: An Overview of Programs and Examples of their Performance", Artificial Intelligence Laboratory, Department of Computer Science, University of Illinois, Urbana IL, Sept. 1986.

R.S. Michalski and J.B. Larson, rev. by K. Chen, "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.

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

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

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

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

K. Spackman, "Integration of Inferential Operators with a Relational Database in an Expert System", Master's Thesis, Department of Computer Science, University of Illinois, Urbana IL, 1982.

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

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

G. Wiederhold, M.G. Walker, R.L.Blum, and S. Downs, "Acquisition of Knowledge from Data", Int. Symp. on Methodologies for Intelligent Systems, Oct. 1986, Knoxville TN.

Zytkow, J.M. (1987) Combining many searches in the FAHRENHEIT Discovery System, Proceedings of the Fourth International Workshop on Machine Learning. Irvine, CA: Morgan Kaufmann, 281-287.

Zytkow, J.M., & Simon, H.A., (1986) A Theory of Historical Discovery: The Construction of Componential Models. Machine Learning, 1, 107-136.