An Integrated Multi-task Inductive Database and Decision Support System VINLEN: An Initial Implementation and First Results

Kenneth A. Kaufman, Ryszard S. Michalski, Jaroslaw Pietrzykowski, and Janusz Wojtusiak

Machine Learning and Inference Laboratory, George Mason University
{kaufman, michalski, jarek, jwojt}@mli.gmu.edu

Abstract. A brief review of the current research on VINLEN multitask inductive database and decision support system is presented. VINLEN integrates a wide range of knowledge generation operators that given input data and/or knowledge create new knowledge. The central operator of VINLEN is a natural induction module that generates hypotheses from data in the form of attributional rules. Such rules are easy to understand and interpret because they directly correspond to equivalent natural language descriptions. This operator is illustrated by an application to discovering relationships between lifestyles and diseases in men. The conclusion outlines plans for future research.

1 Introduction

This paper briefly reviews our current research on the development of VINLEN, a multitask inductive database and decision support system. In VINLEN, inductive inference capabilities are integrated with a database and a knowledge base. Standard relational database operators, implemented through an SQL client, are combined with *knowledge generation operators* (KGOs) using *Knowledge Query Language* (KQL).

KGOs operate on knowledge segments, consisting of a combination of one or more relational tables linked to relevant knowledge component in the VINLEN knowledge base. A KGO takes as input knowledge segments, and generates an output knowledge segment.

Two important capabilities are required from knowledge generation operators: (1) that their results are in a form easy to understand and interpret by users, (2) that KGO-generated results are in forms that can be accepted as input by *compatible* KGO operators. A compatible operator is the one that can use that knowledge if it is submitted to it.

The central knowledge generation operator in VINLEN is implemented in the natural induction module that creates inductive generalizations of or discovers patterns in data in forms that appear natural to people, by employing *attributional calculus* as a representation language [8]. Attributional calculus is a logic system that combines elements of propositional, predicate, and multiple-valued logics for facilitating induc-

tive inference. It serves both as an inference system and as knowledge representation formalism.

Attributional descriptions, the primary form of knowledge representation in VINLEN, are more expressive than conventional decision rules that use only <attribute-relation-value> conditions. Attributional rules use conditions that may involve more than one attribute and relate them to a subset of values or to other attributes. Section 3 gives more details on this topic.

2 An Overview of VINLEN

Research on the VINLEN system grows out of our previous efforts on the development of INLEN, an early system deeply integrating databases and inductive learning capabilities for the purpose of multistrategy learning, data mining, and decision support e.g. [9]. The advantages such integration and of inductive databases are now being widely recognized, as evidenced by this workshop and earlier efforts e.g. [2, 4, 5].

VINLEN represents a step beyond the approach to inductive databases taken by some authors, namely, it not only integrates database and a knowledge base containing selected results of inductive inference (using the capabilities of database), but also is a host of inductive and deductive inference operators, as well as various other data analysis and pattern discovery capabilities. It supports inferences resulting from a series of applications of its operators according to a script in *knowledge query language* (KQL). This way it can automatically conduct experiments that involve creating, storing and managing of the relevant data and knowledge laying the groundwork for higher levels of sophistication in inductive databases functionality based, for example, on a meta-learning approach. Therefore, it provides a powerful tool for conducting experiments and may avoid some pitfalls resulting from too limited exploration of data or from the parameters of the methods [1].

An important concept in VINLEN is that of a *knowledge system* that consists of a database, which can be local or distributed, and a knowledge base. The term "knowledge system" signifies a system integrating a database and a relevant knowledge base to support knowledge mining and knowledge application to the problem at hand. A knowledge base contains handcrafted knowledge and results of applying a range of knowledge generation and management operators to data in the database and/or to prior knowledge in the knowledge base.

The prior knowledge contains definitions of the domains and types of attributes in the database, data constraints, value hierarchies of structured attributes, known relationships binding attributes, and any other background knowledge that users may have represented in the system. During the operation of an inductive database, the knowledge base is populated by newly generated data descriptions, hypothetical patterns, data classifications, statistical information, results from hypothesis testing, etc.

The data in each VINLEN knowledge system are stored internally in relational tables, as are other system components. Most of the entities utilized by the system, such as events (a.k.a. examples), datasets, attributes, attribute domains, rule conditions, rules, rulesets, and classifiers, are presented in individual tables in the database, and connected via relations.

Events are stored in an event table, which is populated either from external source, manually by the user, or by a VINLEN operator, for example, by an operator that selects most representative events from the training dataset. In addition to regular attribute values, events may contain meta-values, such as "unknown", "irrelevant" and "not-applicable" which require a special handling during the learning or knowledge application processes [10]. The "unknown" values, denoted by a "?", represent cases when a regular attribute value exists, but is unknown for some reason, the "irrelevant" and "non-applicable" values represent domain knowledge provided by the user.

Each event may carry additional meta-information, such as *event significanc*e and *event frequency*. The event significance is a value assigned to an event by the user or by the program to represent some form of importance of the event for problem at hand. For different types of problems it may have a different meaning. For example, in concept learning, it may represent the typicality of the event; in optimization via evolutionary computation, it represents the value of the fitness function for that event. Event frequency is a number of occurrences of the given event in the training or testing data.

Prior knowledge and knowledge generated by VINLEN are stored in a hierarchy of relational tables that can be used by KGOs. Rule-based classifiers learned from the data are in the form of families of attributional rulesets. Their components, such as selectors (conditions), complexes (conjunctions of conditions), exceptions, single rules, rulesets, and alternative rulesets are considered as individual entities, and as such are represented by separate tables connected by relations. Parameter sets for individual operators are stored in method-specific tables. This storage methodology facilitates an efficient access to all components of the classifiers through a standard SQL interface.

A *Target Knowledge Specification* is a generalization of a database query; specifically, it is a user's request for a knowledge segment to be created by the system, based on the data and knowledge already present. The core of VINLEN consists of *Inductive Database Operators*, which call upon various programs for knowledge generation (e.g., rule learning, conceptual clustering, target data generation, optimization, etc.), as well as data management. These operators can be invoked by a user directly, or through KQL.

To provide a general overview and easy access to all VINLEN operators, we have developed a visual interface that consists of VINLEN views at different abstraction levels. Fig. 1 presents the most abstract view of the main panel of VINLEN.

The central part contains icons for managing database (DB), knowledge base (KB), and knowledge systems (KS). By clicking on DB, KB, or KS, the user can select and access database, knowledge base, and knowledge system that are available to VINLEN. Each of the rectangular buttons allows the user to access a family of knowledge generation operators of a given type.

For example, the button "Learn Rules" allows one to access operators that learn ordinary attributional rules, rules with exceptions, multi-head rules, and rule-trees. A similar multi-function role have other buttons, such as "Access attributes", "Improve rules", "Learn trees", "Create clusters", "Access scout", "Define dataset", etc.

All operators are integrated through Knowledge Query Language (KQL) that is an extension of the SQL database query language. In addition to conventional data management operators, KQL includes operators for conducting inductive and deductive

inference, statistical analysis, application and visualization of learned knowledge, and various supportive functions. KQL allows a user to define *knowledge scouts* that are KQL scripts for automatically executing a series of knowledge generation operators in search for knowledge of interest to the user.

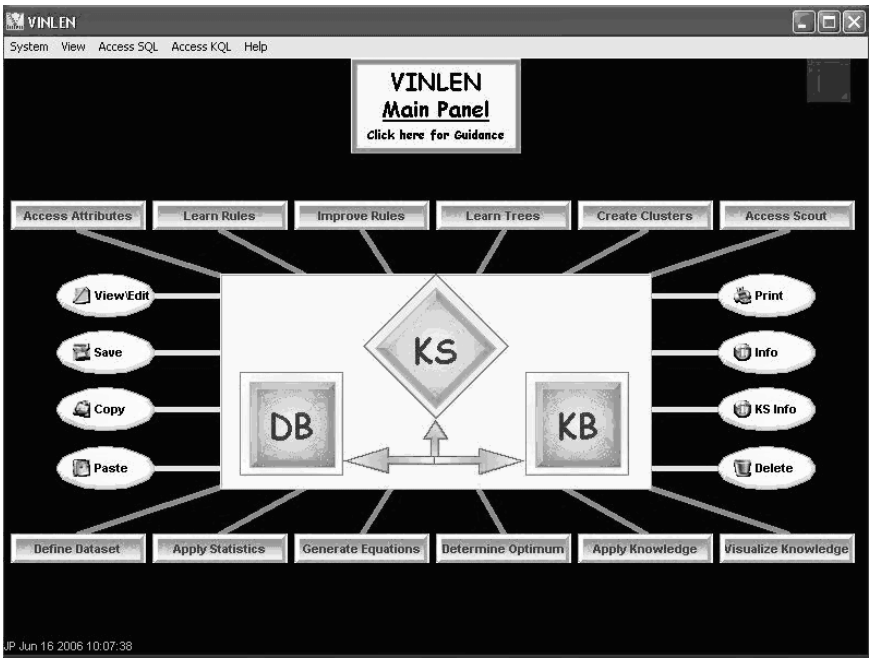

Fig. 1. The front panel of VINLEN (a black and white version of the original)

Due to a wide range of capabilities and the types of operators it involves, many of which are unique to VINLEN, KQL is very different from other high-level languages developed for data exploration, many of which have been Prolog-based.

Among the non-Prolog-based languages, M-SQL [6] is philosophically somewhat similar to KQL in that it builds upon the SQL data query language. It integrates in it, only one inductive operator, in contrast to VINLEN that adds to it a wide range of operators. In [7], the inductive the M-SOL operator is used to analyze users' internet activity. KQML [3] allows the user to query for specific pieces of knowledge, but it does not support multiple discovery operators and the abstract templates that are available in KQL.

A somewhat related to KQL is also a query language presented in [2] that has the capability for specifying the type of knowledge to search for, e.g., rules with the confidence level above a given threshold. Being a Prolog-based language, it has the capability for directly expressing relational descriptions, but does not involve such a wide range and versatile operators that VINLEN does. While VINLEN can also search for rules with a confidence above a certain threshold, it can also seek rules with maximum confidence.

3 VINLEN Operators

As mentioned earlier, VINLEN aims at providing user with an extensive set of different knowledge generation and data management operators, and with a language, KQL, to develop scouts for executing sequences of these operators. Such scouts can thus automatically perform knowledge discovery experiments.

Basic functionality of VINLEN allows the user to browse, edit, copy, delete, print, define, import and export data and knowledge. More advanced functions support data selection, attribute evaluation and selection, attribute discretization, and estimation of parameter settings for operators to achieve a desired result, ,and a range of learning and knowledge discovery functions. The rule learning module is based on the AQ21 program for natural induction [15].

As mentioned earlier, VINLEN's knowledge representation is based on attributional calculus [8]. The basic unit of knowledge representation is an attributional rule in the form:

Consequent <= *Premise* |_ *Exception*,

where *Consequent*, *Premise*, and *Exception* are *conjunctive descriptions*, or *complexes*, which are conjunctions of *attributional conditions*. An attributional condition (a.k.a. *selector*) can be viewed as equivalent to a simple natural language statement. Its general form is:

[*L relsym R*],

where:

L (the left side or referent) contains one attribute, or several attributes joined by "&" or "v", called internal conjunction and disjunction, respectively. L can also be one of the standard *derived attributes*: count, max, min, and avg.

R (the right side or reference) is an expression specifying a value or a subset of values from the domain of the attribute(s) in L. If the subset contains values of a nominal (unordered) attribute, they are joined by the symbol "v" (called internal disjunction); if the subset contains consecutive values of a linear attribute, they are represented by joining the extreme values by operator "..", called range. R can also be a single attribute of the same type as the attribute or attributes in L.

relsym is a relational symbol from the set: {=, ≠, >, ≥, <, ≤}. Relational operators {=, ≠} apply to all types of attributes. Relations {>, ≥, <, ≤} apply only to linear attributes.

Brackets [ ], may be omitted, if their omission causes no confusion. If brackets are used, the conjunction of two selectors is usually written as their concatenation. If an attribute, x, is binary, the condition [x = 1] can be written simply as the literal x, and [x = 0] as the literal ~x. Thus, if attributes are binary, attributional conditions reduce to propositional literals.

An attributional condition is called *basic*, if its left side, L, is a single attribute, the relational symbol is one of { =, >, ≥, <, ≤}, and the right side, R, is a single value; otherwise, it is called *extended*.

Examples of basic conditions:

[x1 = 1], alternatively, x1 (The value x1 is 1)
[x1 = 0], alternatively, ~ x1 (The value x1 is 0, the alternative notations assume that x1 is binary)
[color = red] (The color is red)
[length < 5"] (The length is smaller than 5 inches)
[temperature ≥ 32° C] (The temperature is greater than or equal to 32° C)
[tools={mallet, knife}] (The tools are mallet and knife)

Examples of extended conditions:

[color = red v blue v green] (The color is red, blue or green)
[blood-type ≠ A] (The blood type is not A)
[length= 4..12] (The length is between 4 and 12, inclusive)
[color ≠ green] (The color is not green)
[height > width] (The height is greater than the width)
[height v width < 3 m] (The height or the width is smaller than 3 m)
[height & width ≥ 7 cm] (The height and width are both at least 7 cm)
[height & width < length] (Both the height and the width are smaller than the length)

Operators "v" and "&" when applied to non-binary attributes or to their values are called internal disjunction and conjunction, respectively.

A set of attributional rules with the same consequent (e.g., indicating the same class) is called an *attributional ruleset*. A set of attributional rulesets whose consequent spans all values of an output (dependent variable) is called an *attributional classifier*.

The design of VINLEN includes the following learning and inference capabilities:

- Learning complete and consistent attributional classifiers
- Optimizing attributional classifiers
- Discovering strong patterns in data (attributional rules that represent strong regularities but may be partially inconsistent with the data)
- Generation of multi-head attributional rules (with more than one attribute in the consequent of a rule);
- Deriving optimized decision trees from attributional classifiers e.g. [13];
- Applying attributional classifiers to data, and evaluating the results in the case of testing data;
- Discovering conceptual clusters in data e.g. [11];
- Determining the optimum of a given function using non-Darwinian evolutionary computation [14];

To facilitate the interpretability and understandability of learned knowledge, VINLEN includes operators that visualize knowledge in the form of *concept association graphs* and *generalized logic diagrams* e.g. [12].

It should be mentioned that although all the knowledge generation operators described above have been developed, implemented, and tested individually, so far only some of them have been fully integrated in VINLEN. The process of integration of so many operators, and developing an appropriate graphical user interface for each them is a very labor-intensive effort, and it will take some time before it is completed.

4 Knowledge scouts

As mentioned earlier, VINLEN operators (learning and inference operators, as well as data and knowledge management operators) can be arranged into knowledge scouts. A knowledge scout is a KQL script that for automatically applying various operators in search of *target knowledge* in the database. The target knowledge is defined abstractly by specifying properties of pieces of knowledge that are of interest to the given user (or a specified group of users). Simple examples of target knowledge specification are "Determine a general classifier from the dataset DS, that maximizes the criterion of attributional classifier ACQ", or "Determine a conceptual clustering of the dataset DS that optimizes the clustering quality criterion CCQ."

Here is very simple example of a knowledge scout in the form of a one-line KQL script:

```
CREATE RULES GDP-Classifier TYPE CC FROM TR1 USING AQ21
WHERE decision is "GDP", searchscope = 3
```

This script instructs VINLEN to apply a rule learning operator to the data set TR1, using the AQ21 module with the *searchscope* parameter set to 3 (the width of the beam search used in the learning module). Since settings of other parameters are not specified, the default values will be used. The goal of applying the learning operator is to learn a consistent and complete (CC) classifier for the output attribute "GDP" The classifier is to be stored in the knowledge base under the name "GDP-Classifier."

In order to synthesize target knowledge, a knowledge scout may consist of many lines of KQL code that request an execution of a sequence of KQL operators involving data, intermediate results, previously learned knowledge and background knowledge. The latter may include the types of attributes, their domains (including hierarchies of structured attributes), problem constraints, and rules for constructing derived attributes. At every step of running knowledge scout, an application of one operator may depend on the results of previous operators. This is possible due to the inclusion of tests of properties of data and knowledge components, of the results of their application to data, and the use of a branching operator in KQL. For example, a condition for repeating a learning operator may be:

"If the average consistency of attributional rules in the classifier is smaller that .95 or the number of rules in the classifier is greater than 10, the accuracy of the classifier

on the testing data is smaller than .93, and the number of learning runs is smaller than 50, repeat the run with the *searchscope* 15, otherwise, return the results."

5 An Example of Application to a Medical Domain

This section illustrates an application of the learning module of VINLEN to a problem of determining relationships between lifestyles and diseases of non-smoking males, aged 50-65, and displaying results in the form of a concept association graph. The study employed a database from the American Cancer Society that contained 73,553 records of responses of patients to questions regarding their lifestyles and diseases. Each patient was described in terms of 32 attributes: 7 lifestyle attributes (2 Boolean, 2 numeric, and 3 rank), and 25 Boolean attributes denoting diseases. The learning operator determined patterns (approximate attributional rules) characterizing the relationships between 25 diseases and the lifestyles and other diseases. Fig. 2 shows one example of discovered patterns (*HBP* stands for High Blood Pressure, *Rotundity* is a discretized ratio of the patient's weight to his height and *YinN* denotes the years the patient lived in the same neighborhood).

[Arthritis = Present] <=
[HBP=present: 432, 1765] &
[Education<=college_grad: 940, 4529] &
[Rotundity>=low: 1070, 5578] &
[YinN > 0: 1109, 5910]: p = 325, n = 1156; P = 1171, N = 6240

Fig. 2. A pattern for Arthritis discovered in the medical database.

The two numbers listed within each condition after the colon denote the numbers of positive and negative examples in the training set covered by that condition, respectively; p and n, are the numbers of positive and negative examples in the training set covered by the entire rule, respectively; and P and N are the numbers of positive and negative examples in the training data for that class (here, Arthritis), respectively.

The pattern in Fig. 2 defines a set of conditions under which patients had arthritis relatively frequently. These conditions include the presence of high blood pressure, no education beyond college, more than "very low" rotundity, and the patient's having lived in his current neighborhood at least one year. In the training data, about 16% of the patients had arthritis (P / (P + N)), but among patients satisfying the pattern, the percentage grows to 22% (p / (p + n)). The most significant factor in the pattern is high blood pressure, which by itself has consistency of about 19%. The discovered patterns were visualized using a concept association graph. Fig. 3. presents one such graph that was automatically generated using the CAG visualization operator.

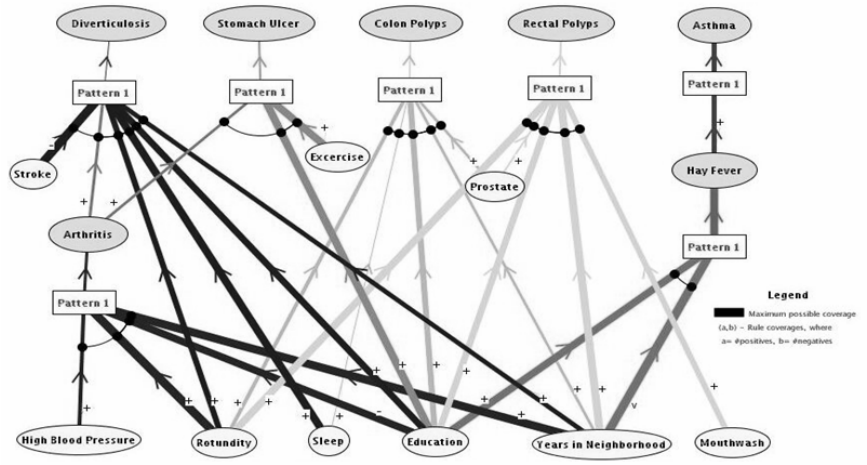

Fig. 3. Concept Association Graph representing seven patterns in the medical database.

The thickness of links in CAG reflects the condition's consistency, and its annotation (+, –, v, or ^) indicates the type of the relationship between condition and consequent. Specifically, "+" and "-" represent a positive and negative monotonic relationship, respectively; and "v" and "^" indicate that extreme values of the attribute indicate higher or lower values of the consequent attribute, respectively.

While no claim is made as to the medical validity and significance of these results, they indicate, however, that the developed methodology is potentially capable of discovering important patterns in the data and representing them in an understandable way, either as qualitative relationships in the form of association rules, or graphically via a concept association graph.

6 Summary and Future Work

The paper described current research on VINLEN, a large-scale system for integrating operators for data management, data analysis, knowledge discovery for classification, clustering and optimization, for data and knowledge visualization, knowledge testing and application, and decision support. The underlying knowledge representation is based on attributional calculus that combines features of propositional, predicate, and multi-valued logic for the purpose of facilitating knowledge discovery.

Individual operators can be invoked by the user individually, via graphical user interface, or automatically, via a knowledge scout, which is a script in high level language, called knowledge query language (KQL). KQL is an extension of SQL that adds to it various knowledge generation, management, visualization and application operators.

The system is still under development. In this paper, we focused on two central operators already implemented in VINLEN, namely, the operator for learning attributional classifiers, and the operator for visualizing such classifiers using concept asso-

ciation graphs. These operators have been illustrated by an example in medical domain.

Other operators, such conceptual clustering, intelligent target data and generation, function optimization via Learnable Evolution, and database manipulation through an SQL client have been partially implemented. Various statistical operators, modules for applying knowledge to data for generating decisions, and a mechanism for creating knowledge query language scripts to guide data exploration tasks are under development.

The main contribution of the VINLEN project is the development of a general methodology for a tight integration of a database, knowledge base, data management operators, knowledge generation operators, a knowledge query language, and a user-oriented visual interface. Research on VINLEN aims at achieving methodological advances and for developing new tools for knowledge mining in databases, and for the decision support based on the knowledge discovered.

Acknowledgments

Research described here has been conducted in the Machine Learning and Inference Laboratory at George Mason University and has been supported in part by the National Science Foundation under Grants No. IIS-9906858 and IIS-0097476, and in part by the UMBC/LUCITE #32 grant. In a few cases, presented results have been obtained under earlier funding from the National Science Foundation, the Office of Naval Research, or the Defense Advanced Research Projects Agency. The findings and opinions expressed here are those of the authors, and do not necessarily reflect those of the above sponsoring organizations.

References

1. Blockeel, H.: Experiment Databases: A Novel Methodology for Experimental Research. In: Bonchi, F., Boulicaut, J-F. (eds.): Knowledge Discovery in Inductive Databases. 4th International Workshop, KDID'05, Revised, Selected and Invited Papers. Lecture Notes in Computer Science, Vol. 3933. Springer-Verlag, Berlin Heidelberg New York (2006) 72-85
2. De Raedt, L.: A Perspective on Inductive Databases. ACM SIGKDD Explorations Newsletter, Vol. 4, Issue 2. ACM Press, New York, NY, USA (2002) 69-77
3. Finin, T., Fritzson, R., McKay, D. and McEntire, R.: KQML as an Agent Communication Language. Proceedings of the Third International Conference on Information and Knowledge Management, CIKM'94. ACM Press (1994) 456-463
4. Flach, P. and Dzeroski, S.: Editorial: Inductive Logic Programming is Coming of Age. Machine Learning, Vol. 44, Number 3. Springer Netherlands (2001) 207-209
5. Hätönen, K. Mika Klemettinen, M., Miettinen, M.: Remarks on the Industrial Application of Inductive Database Technologies. In: Boulicaut, J-F., De Raedt, L., Mannila, H. (eds.): Constraint-Based Mining and Inductive Databases, European Workshop on Inductive Databases and Constraint Based Mining, Hinterzarten, Germany, March 11-13, 2004, Revised Selected Papers. Lecture Notes in Computer Science, Vol. 3848. Springer (2005) 196-215
6. Imielinski, T., Virmani, A. and Abdulghani, A.: DataMine: Application Programming Interface and Query Language for Database Mining. In: Proceedings of the 2nd International

Conference on Knowledge Discovery and Data Mining, KDD'96. AAAI Press (1996) 256-261
7. Meo, R., Vernier,F., Barreri, R., Matera, M. and Carregio, D.: Applying a Data Mining Query Language to the Discovery of Interesting Patterns in WEB Logs. Workshop on Inductive Databases and Constraint Based Mining, Hinterzarten, Germany (2004)
8. Michalski, R.S.: ATTRIBUTIONAL CALCULUS: A Logic and Representation Language for Natural Induction. Reports of the Machine Learning and Inference Laboratory, MLI 04-2, George Mason University, Fairfax,, VA (April 2004)
9. Michalski, R.S., Kerschberg, L., Kaufman, K. Ribeiro, J.: Mining For Knowledge in Databases: The INLEN Architecture, Initial Implementation and First Results. Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies, Vol. 1, No. 1 (August 1992) 85-113
10. Michalski, R.S., Wojtusiak, J.: Reasoning with Meta-values in AQ Learning. Reports of the Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA (June 2006)
11. Seeman, W.D., Michalski, R. S.: The CLUSTER3 System for Goal-oriented Conceptual Clustering: Method and Preliminary Results. Proceedings of The Data Mining and Information Engineering Conference, Prague, Czech Republic, July 11-13, 2006.
12. Sniezynski, B., Szymacha, R., Michalski, R. S.: Knowledge Visualization Using Optimized General Logic Diagrams. Proceedings of the Intelligent Information Processing and Web Mining Conference, IIPWM 05, Gdansk, Poland, June 13-16, 2005.
13. Szydlo, T., Sniezynski, B., Michalski, R. S.: A Rules-to-Trees Conversion in the Inductive Database System VINLEN. Proceedings of the Intelligent Information Processing and Web Mining Conference, IIPWM 05, Gdansk, Poland, June 13-16, 2005.
14. Wojtusiak, J., Michalski, R. S.: The LEM3 Implementation of Learnable Evolution Model and Its Testing on Complex Function Optimization Problems. Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2006, Seattle, WA, July 8-12, 2006.
15. Wojtusiak, J., Michalski, R. S., Kaufman, K., Pietrzykowski, J.: Multitype Pattern Discovery Via AQ21: A Brief Description of the Method and Its Novel Features. Reports of the Machine Learning and Inference Laboratory, MLI 06-2, George Mason University, Fairfax, VA (2006)