Fundamenta Informaticae 40, pp. 433-447, 2000.

Building Knowledge Scouts Using KGL Metalanguage

Ryszard S. Michalski* and Kenneth A. Kaufman
Machine Learning and Inference Laboratory
George Mason University
Fairfax, VA 22030-4444, USA

* Also with Institute of Computer Science,
Polish Academy of Sciences, Warsaw, Poland

{michalski,kaufman}@gmu.edu

Abstract

Knowledge scouts are software agents that autonomously search for and synthesize user-oriented knowledge (*target knowledge*) in large local or distributed databases. A knowledge generation metalanguage, KGL, is used to creating scripts defining such knowledge scouts. Knowledge scouts operate in an *inductive database*, by which we mean a database system in which conventional data and knowledge management operators are integrated with a wide range of data mining and inductive inference operators. Discovered knowledge is represented in two forms: (1) *attributional rules*, which are rules in *attributional calculus*—a logic-based language between propositional and predicate calculus, and (2) *association graphs*, which graphically and abstractly represent relations expressed by the rules. These graphs can depict multi-argument relationships among different concepts, with a visual indication of the relative strength of each dependency. Presented ideas are illustrated by two simple knowledge scouts, one that seeks relations among lifestyles, environmental conditions, symptoms and diseases in a large medical database, and another that searches for patterns of children's behavior in the National Youth Survey database. The preliminary results indicate a high potential utility of the presented methodology as a tool for deriving knowledge from databases.

Keywords: data mining, knowledge discovery, knowledge scouts, inductive databases, knowledge visualization, knowledge generation language, association graphs, attributional calculus.

1 Introduction

When applying data mining tools to a large database, a user may have to perform many repetitions and trials of various operations before desired knowledge (target knowledge) is determined. This process can be particularly difficult and time-consuming if the data mining system includes many different data mining operators/tools that can be applied to the database. Such a situation occurs, for example, in the multistrategy data mining system INLEN (Michalski, 1997; Michalski and Kaufman, 1998).

Another problem in data mining is how to specify *target knowledge*, that is, knowledge that is likely to be of interest to a given user or a group of users. Obviously, such knowledge cannot be defined precisely, as the whole purpose of the search is to find something new and unexpected. An additional problem is that the target knowledge may be changing over time, as it depends on the current goals and current knowledge of the user. The latter indicates a need for a mechanism that is able to acquire and monitor the profile of the user's interests, and apply this profile in the search for target knowledge.

To address problems outlined above, the idea of a *knowledge scout* is proposed. A knowledge scout is a software agent that employs resources of an *inductive database* for automatically searching for and synthesizing target knowledge. By an inductive database we mean a system that integrates a conventional database with inductive inference capabilities. Such capabilities allow a database to answer queries asking for *plausible knowledge*, that is, knowledge that is not directly or deductively obtainable from the database, but can be hypothesized through inductive inference for the data. Such knowledge can be in the form of hypotheses about future datapoints, expected consequences from the data, generalized data summaries, emerging global patterns, exceptions from hypothesized patterns, suspected errors and implied inconsistencies, hypothetical plans synthesized from the data, etc. (Michalski, 1983; Han et al.,1996; Imielinski, Virmani, and Abdulghani, 1996; Michalski, 1999). A general diagram of an inductive database is presented in Figure 1.

An inductive database implements new types of database operators that are based on methods for inductive inference developed in the fields of machine learning, statistics, and uncertain reasoning. These operators, together with conventional database operators, are integrated into a single *knowledge generation language (KGL).* An inductive database includes a *knowledge base* that may contain meta-knowledge, domain constraints, specifications of attribute types and domains, models of users' interests, knowledge obtained in previous data mining operations, etc. Using KGL, one can implement different knowledge scouts, dedicated to pursuing various target knowledge. A KGL

script that defines a knowledge scout includes a plan of operations to be performed on the database (local or distributed), and an abstract definition of the target knowledge.

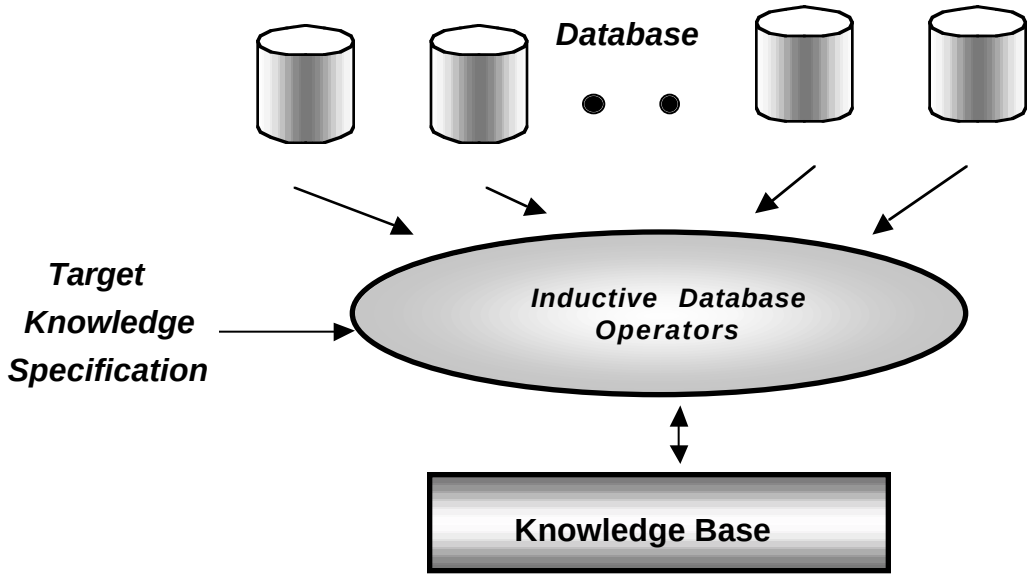

Figure 1. A general diagram of an inductive database.

The target knowledge for a scout is defined abstractly by specifying properties of pieces of knowledge that are likely to be of interest to the given user (or a group of users). For example, a target knowledge may be defined as "patterns that relate variables from the set T to those in the set S", or "patterns that achieve the highest score on a given pattern quality measure" (e.g., Kaufman and Michalski, 1999), or "a classification of data that maximizes a criterion of clustering quality" (e.g., Michalski and Stepp, 1983).

In order to synthesize target knowledge, a knowledge scout may execute long sequences of different kinds of operations involving data, intermediate results, and background knowledge from the Knowledge Base. At every step of this process, an application of one operator may depend on the results of previous operators. The user's interests and past relevant knowledge are partially defined a priori, and partially constructed and updated during the scout's lifetime. A knowledge scout can operate for a specific period of time, or work continuously in the background, outputting its findings when some predefined condition is met (*alert condition*) or upon the user's request. As inductively derived knowledge generally has lower certainty than directly or deductively obtained knowledge, results of inductive queries are annotated by certainty measures (for example, by the testing accuracy of the learned rules).

To implement the above capabilities, one needs a knowledge representation system for expressing knowledge to be synthesized, and a knowledge generation language for defining knowledge scouts. These issues are addressed in the following three sections. Section 2 describes a knowledge representation system based on attributional rules, Section 3 describes association graphs for graphically representing the rules, and Section 4 describes KGL-1, our initial knowledge generation language.

2 Attributional Rules

The language in which patterns or knowledge of interest are to be expressed is essential to the ability to discover them. If the language is too restricted, patterns will have complex expressions, and this, in turn, will make their discovery difficult. If the language is too rich, the pattern search space may become computationally prohibitive. In addition, for many applications it is important that patterns are easy to understand and interpret (the comprehensibility postulate; Michalski, 1983). Guided by such considerations, we employ *attributional calculus rules* for expressing patterns or knowledge of interest (Michalski, 1999).

The attributional calculus is an extension of propositional calculus in which literals (propositions and their negations) are replaced by *attributional conditions*. Such conditions represent relational statements that bind one or more attributes with a set of their values or other attributes (see below). Each attribute has a domain and a type; the former defining its set of legal values, and the latter characterizing an ordering relationship among the values. Attributional calculus is based on variable-valued logic system VL1 (Michalski, 1975).

An attributional condition is expressed in the form: **[L rel R]**, where **L** (*left side*) is an attribute, or one or more attributes with the same domain joined by "&" or "v" (these operators are called *internal conjunction* and *internal disjunction*, respectively); **R** (*right side*) is a value or a list of values joined by the symbol "v" or the word "or" (called *internal disjunction*), a pair of values joined by ".." (called *range*), or an attribute with the same domain as the attribute(s) in **L**; and **rel** is a relational symbol from the set $\{=, \neq, >, \geq, <, \leq\}$. A condition **[L rel R]** is *true* (or *satisfied*), if expression **L** is in relation **rel** to **R**. For illustration, the following are examples and explanations of attributional conditions. Note that attributional conditions are simple to interpret and easy to translate to equivalent natural language expressions.

[blood-pressure = normal] *(the blood pressure of the patient is normal)*
[income = 20K..30K] *(the income is between 20K and 30K)*
[color = red v blue] *(the color is red or blue)*
[width & length > depth] *(the width and length are both greater than the depth)*

Attributional rules used in this study are in the form **<decision> if <conditions**>, where **<decision>** is a single attributional condition, and **<conditions>** is a conjunction of one or more attributional conditions. These rules are a special case of the *parameterized association rules* (PARs), described in (Michalski, 1989). The association rules presented in (Agrawal, Imielinski and Swami, 1993) could be viewed as a specialized form of PARs.

Attributional rules that characterize a pattern in a database can be determined using an inductive operator based on the AQ-18 rule learning program (Kaufman and Michalski, 1999). Such an operator generates rules with annotations specifying the *support*, *disparity*, *completeness* and *consistency* for each condition in the rule, and for each rule as a whole (the *if-part*). The support of a condition [or, a rule], denoted by *p*, is defined as the number of tuples representing a given relationship ("positive examples") that satisfy the condition (the rule).

The disparity, denoted by *n*, is defined as the number of "negative examples" that satisfy the condition (the rule). The completeness, denoted *compl*, is defined as p / P, where P is the total number of training examples in the positive class. The consistency, denoted *cons*, is defined as (p / $p + n$). The program also generates other annotations, such as exceptions, ambiguity, rule quality, which are described elsewhere (e.g., Kaufman and Michalski, 1999).

The following example illustrates one of the attributional rules generated by a knowledge scout seeking demographic patterns in a World Factbook database (in a somewhat simplified form; see Section 4). For this experiment, countries of the world were divided into classes representing different fertility rate ranges. The rule in Figure 2 characterizes 16 of the 42 countries with the smallest fertility rates (no more than 2 per woman).

Fertility ≤ 2 per woman if:	*p*	*n*	*compl*	*cons*	
[Birth Rate = *10..20 per 1000 people*]		42	20	100%	68%
[Religion is *R. Catholic* or *Orthodox* or *Anglican* or *Shinto*]		24	31	57%	44%
[Infant Mortality Rate ≤ *40 per 1000 babies*]		41	54	98%	43%
[Population Growth Rate ≤ *4%*]		32	56	76%	36%
[Literacy ≥ *70%]*		35	71	83%	33%
[Life Expectancy = *60..80 years*]		41	92	98%	31%
[Death Rate = *5..15 per 1000 people*]		42	102	100%	29%
[Net Migration Rate ≥ *-10 per 1000 people*]		42	140	100%	23%
Rule Total (all conditions):		*16*	*0*	*38%*	*100%*

Figure 2. Example of an annotated rule in the attributional calculus.

3 Association Graphs

Attributional rules provide details about relationships among attributes or concepts. To illustrate such relations graphically and abstractly, we developed a visualization method called *association graphs*. In an association graph, nodes represent attributes or concepts, and inter-node links characterize relationships among nodes. The links are directed, weighted and annotated. The direction of a link indicates the direction of the relationship. The weight of a link is represented by the thickness of the link. The thicker the link, the stronger the relationship (as specified by the consistency of the attributional condition). Links are annotated by symbols indicating the type of relationship between connected nodes. A monotonically growing (decreasing) functional relationship between variables is indicated by the symbol "+" ("–") attached to the link between corresponding nodes.

A functional relationship that has its maximum (minimum) in the middle of the range of the independent (input) attribute is indicated by the symbol "^" ("v"). Links that represent relationships that do not fall to any of the above types are left unlabeled. These symbols are also used when the relationship only approximates one of the relationship classes defined above. A rule relating several attributional conditions to another condition is represented by linking the involved conditions with an arc. For example, Figure 3 shows an association graph representing the rule from Figure 2.

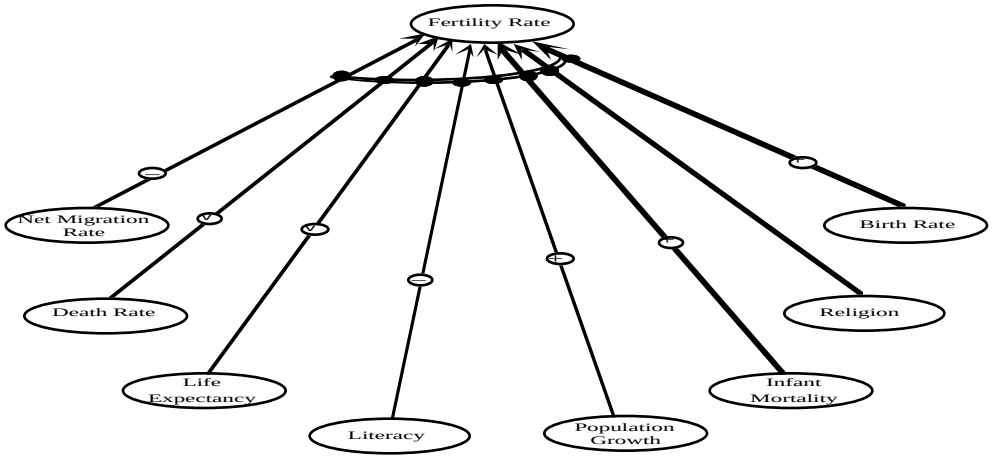

Figure 3. An association graph representing the attributional rule from Figure 2.

Association graphs can simply represent complex multivariate relationships. They provide a more advanced tool for knowledge visualization than that used in some data mining systems (e.g., in CLEMENTINE, a data mining toolkit commercially developed by Integral Systems, Ltd.). One major difference is that association graphs can represent multi-argument relations, not only binary relations, as in CLEMENTINE. Another difference is that the association graphs are

representations at a higher abstraction level. Specifically, their nodes represent attributes, rather than individual attribute values, and links represent composite conditions employed in attributional calculus, rather than only attribute-value conditions.

4 A Metlanguage for Defining Knowledge Scouts: KGL-1

Knowledge scouts are defined by creating scripts in the knowledge generation language (KGL). Below is a brief description of our first version of such a language, called KGL-1 (Kaufman and Michalski, 1998). KGL-1 has been designed according to the following requirements:

1. The language integrates database operators, knowledge base operators, and knowledge generation operators in a single representational system.
2. Inductive inference programs, as well as other knowledge processing programs integrated in the inductive database can be invoked by single KGL-1 operators.
3. Results from any KGL-1 operator can be used as inputs to any operator for which they are semantically applicable.
4. Parameters to be used in running any knowledge-generating program can be specified as arguments of the corresponding KGL-1 operator.
5. KGL-1 statements can refer to various properties of the data in the database (For example, "If there are 10% new examples of class A in the database, invoke a rule learning operator"; "Determine the percentage of missing values in the database.")
6. KGL-1 statements can refer to various properties of generated knowledge or the background knowledge, in particular, to an attribute or attribute values, to the type and the domain of any attribute, the attributional rules and their components, to the groups of rules (rulesets), and to any component of the annotations of the rules, etc. (For example, "Select nominal attributes with five or fewer values in their domain as dependent variables, and generate for them decision rules that use only numerical attributes as an independent variables," "If the completeness of a rule in a ruleset for a given class is at least 95%, remove from the ruleset all rules with a lower completeness," or "Determine all conditions in a ruleset for a given class whose consistency is above 80%, support is between 30 and 50, and **R** part contains only single values.")
7. Looping and branching are implemented as in conventional programming languages.
8. The language has capabilities for invoking any data management, knowledge management and knowledge generation operators that may be involved in the extraction, manipulation, generation and displaying of any data or knowledge in the system. (For example, a script can require to select as the target data a dataset containing attributes from a set A, generate attributional

rulesets for each value of each attribute in A, select the best rules according to a given quality criterion, and then display the rules using a rule visualization operator.)

KGL-1 has been partially implemented in the INLEN-3 system (Michalski and Kaufman, 1998). Each operator specifies an operation on the database and/or knowledge base, and contains arguments that identify its input, output, and the parameters (Params) that deviate from the defaults. Parameters of each operator make it possible to specialize it to usually several different forms, thus each operator corresponds in reality to a set of different data and/or knowledge transformations. The input data to an operator does not have to be specified, if it is supposed to use data obtained from the last operator that determined that data applicable to this operator.

The following operators have already been integrated into the language, or are in the process of integration through the adaptation of already implemented programs:

CHAR(Datatable, Class, Params): Characterize a set of entities in the Datatable that belong to the Class, by inducing their characteristic description (Michalski, 1983).

DIFF(Datatable, Class1, Class2, Params): Differentiate entities in Class1 from Class2 in the Datatable, by inducing a discriminant description (Michalski, 1983).

SELECT(Target, Datatable, Params): Select components from the Datatable, whose type is defined by the Target. The Target can be "attributes", in which case the Datatable is projected on attributes selected according to the method specified in Params. Such a method may just call for a selection of attributes designated by the program (or a user), or may invoke an inference program that seeks the most relevant attributes for a given task. The Target can also be "examples", in which case the SELECT operator selects a subset of records from the Datatable according to the method specified in Params. Such a method may just call for a selection of records designated by the program (or a user), or may invoke a program that seeks the most representative records (examples) for a given task.

TEST(Datatable, Ruleset, Params): Test the knowledge contained in the Ruleset against a set of testing examples in the Datatable. Each testing example is classified based on the rule it best matches, using a strict or a flexible matching method (Reinke, 1984; Bergadano et al., 1992). The operator generates a report on the ruleset's predictive accuracy, and shows how each example was classified.

CLASSIFY(Examples, Ruleset, Params): Assign the Examples (can be a single example) to corresponding classes using the Ruleset. This operator invokes an inference procedure that applies rules in the Ruleset to Examples. The output of this operator includes a list of records from EXAMPLES, their classification and some measure of certainty that that classification.

CLUSTER(Datatable, Params): Split records in the Datatable into a set of conceptual clusters. The operator is based on the conceptual clustering program CLUSTER2 (Michalski and Stepp, 1983; Fischthal, 1997). It defines clusters by attaching a column to the Datatable with indices indicating clusters, and describes each cluster by an attributional rule.

GENSTAT(Datatable, Params): Determine and report statistical characteristics of the Datatable, such as means, modes and variances for attributes in subsets of data associated with different target variables. It can also generate covariances and correlation coefficients between attributes.

VISUALIZE(Input, Params): Visualize the datatable and/or attributional rules specified in the Input, using a diagrammatic visualization method (Zhang and Michalski, 1999).

To illustrate how KGL-1 can be used for building knowledge scouts, Figure 4 presents a KGL-1 script for a simple knowledge scout that creates and examines a knowledge base of relationships between each attribute in the World Factbook database, and all the remaining attributes. Each such relationship is expressed by a set of attributional rules, generated by the Char operator. One of these rules was illustrated in Figure 2.

```
open PEOPLE                                    {Select PEOPLE database}
do CHAR(decision=all, pfile=people1.lrn)       {Characterize concepts
                                                representing single values of
                                               all attributes, using parameters
                                               specified in file people1.lrn}
strongPGrules1 = #rules(PGR, compl >= 60)      {Count rules for Population}
strongPGrules2 = #rules(PGR, supp >= 25)       {Growth Rate that satisfy}
strongPGrules3 = #rules(PGR,                   {three different conditions}
   num_conds(cons >= 50% and supp > 10) > 2)   {for threshold of strength}
print "Number of strong PGR rules:
   Type 1 = ", strongPGrules1, ",
   Type 2 = ", strongPGrules2, ",
   Type 3 = ", strongPGrules3
if #conditions(Fert) > 150                     {Is Fert ruleset too complex?}
  begin
  do SELECT(attributes, decision=Fert,
     thresh=4, out=PEOPLE2, criterion=max)     {If so, find "thresh" best}
  do CHAR(pfile=people1.lrn, decision=Fert)    {independent attributes, then
  end                                          {recharacterize}
for i = 1 to 6
begin                                          {For each value of i from 1 to}
print "Number of LE conditions with p/n       {6, count and display number of}
          ratio of at least", i, ":1 =",       {Life Expectancy conditions with}
          #conditions(LE, cons >= i/(i+1))     {consistency ≥ i/(i+1).}
end
```

Figure 4. A KGL-1 script for defining a knowledge scout exploring a demographic database.

The log file output from the above script is shown in Figure 5. The first part of the output shows the number of strong attributional rules, as determined by three different criteria imposed on the

rule strength (see script annotations). The first two criteria are self-explanatory. The third criterion accepts attributional rules in which the number of conditions with consistency not smaller than 50% and with support greater than 10 is more than two). Because the Fertility ruleset was found too complex (having more than 150 conditions), a learning process was repeated using only the four most relevant independent attributes, as determined by the operator SELECT. The last part of the output presents numbers of conditions in the ruleset for Life Expectancy that exceed different thresholds regarding the *p / n* ratio (that is, support divided by disparity, assuming that the disparity is not zero; when disparity is zero, all thresholds are assumed to be satisfied). The last three lines in the output indicate that there was only one condition with p/n ratio greater or equal 4:1 in the Life Expectancy ruleset.

```
Number of Strong PGR rules: Type 1 = 1, Type 2 = 1, Type 3 = 7
Selecting best attributes from PEOPLE for concept Fert -- Attributes chosen:
          Birth Rate, Predominant Religion, Life Expectancy (LE), Death Rate
Number of LE Conditions with p/n ratio of at least 1:1 = 25
Number of LE Conditions with p/n ratio of at least 2:1 = 10
Number of LE Conditions with p/n ratio of at least 3:1 = 5
Number of LE Conditions with p/n ratio of at least 4:1 = 1
Number of LE Conditions with p/n ratio of at least 5:1 = 1
Number of LE Conditions with p/n ratio of at least 6:1 = 1
```

Figure 5. Output from the KGL fragment from Figure 4.

The KGL-1 meta-language presented above provides a unique combination of features, which is not present in other languages for automated knowledge discovery. Most current languages use a Prolog-based approach, and have quite limited types of knowledge generation operators available. One of the exceptions to the Prolog-based approach is M-SQL, which extends the SQL data query language by adding to it the ability to query for certain types of rules and to invoke an association rule generating operator (Imielinski, Virmani, and Abdulghani, 1996). KGL-1 differs from M-SQL in that it is able to define complex data mining plans that may involve many different types of knowledge generation operators, and more closely resembles a programming language than a query language.

A language somewhat related to KGL-1 is KQML, which provides means by which agents may communicate among themselves and exchange information needed to complete their individual tasks (Finin et al. 1994). Another related language is used in CLEMENTINE, which allows a user to specify a plan for a sequence of actions by a simple interface.

Summarizing, KGL-1 supports a powerful attributional calculus representation, employs a wide range of symbolic learning and inference operators, allows operations on various components of attributional rules, the knowledge base and the database, and provides mechanisms for

implementing advanced knowledge scouts. The attributional rules allow the system to compactly and understandably represent complex multidimensional relationships.

5 Study 1: A Knowledge Scout for Determining Relationships in a Medical Database

Among major means for improving medical decision making and preventing diseases is to develop advanced models of the relationships among medical conditions, manifestations, lifestyles, and therapies. Such models must be able to represent multidimensionality of relations, for example, that a confluence of several factors may be needed to develop a given disease.

Our first efforts toward building such models involved developing a knowledge scout that searches for strong patterns in a database representing facts about diseases, manifestations and lifestyles, developed by the American Cancer Society's Second Cancer Prevention Study (CPS-II). For our preliminary experiments, we selected a small subset of records (73,553 records from 1.2 million), which pertains to male non-smokers, ages 50-65.

Each patient was characterized in terms of about 30 attributes, such "rotundity" (a function of the patient's height and weight), the amount of exercise, the number of hours of sleep, the education level, the use of mouthwash, etc. Together with these characteristics was information whether or not he had occurrences of any of the 25 types of disease. The study involved building a knowledge scout for determining strong patterns that link patients' characteristics and lifestyles with individual diseases. Over 10,000 patterns were generated. A small collection of them is presented in Figure 6 using an association graph.

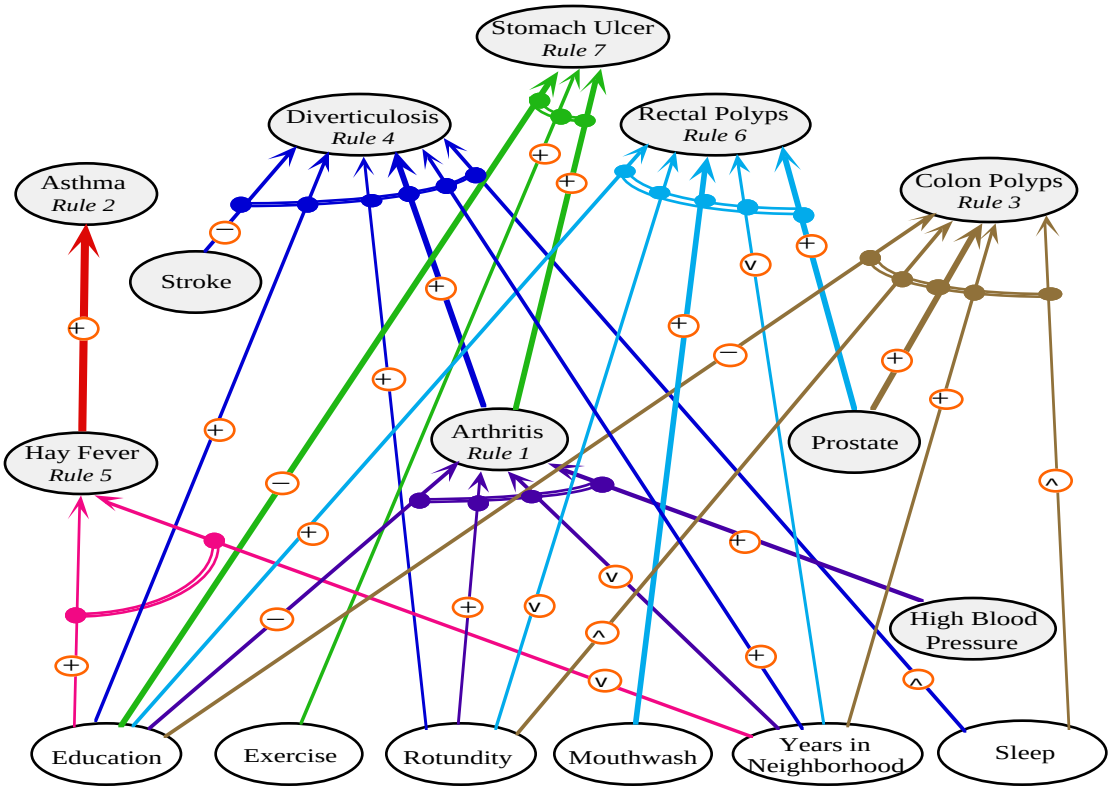

Figure 6. An association graph linking a group of diseases with patient characteristics determined from a subset of the ACS Second Cancer Prevention Study database.

This study was preliminary, and its results should not be taken as new medical knowledge. However, these early results show a significant potential of the proposed methodology for discovery a useful knowledge. Association graphs such as the one in Figure 6 can provide new insights into relationships between diseases and lifestyles, and assist doctors in the disease diagnosis and treatment. They can also serve as guides to patients for disease prevention.

6 Study 2: A Knowledge Scout for Determining Patterns in Parent-Child Database

This study concerned building a knowledge scout for exploring the National Youth Survey database (Elliott, 1976). Children from across the United States were interviewed along with their parents or other adult guardians. The dataset contained 1735 records, each characterizing one child in terms of about 600 multi-valued attributes. About 30% of the attributes represented parents' responses, 65% represented children's' responses, and the rest represented environmental factors.

A knowledge scout was created by writing a KGL-1 script for generating attributional rules for each value (decision class) of all non-metric attributes (415 decision attributes, each of which had from 2 to 9 values). For each decision attribute, as independent variables were used 25 attributes, which were determined as the most relevant to the given task by the SELECT operator. From the

generated rulesets, the scout determined strong attributional rules, defined as those whose completeness was over 80%, or whose completeness was over 60% and support was over 50 cases.

```
begin
for i = 0 to 414
  begin
  open SURVEY
  do SELVAR(decision=i, out=SURVEY2, thresh=25, criterion=avg)
  do DISCSET(decision=0, scope=1, compile=no)
  end
end
open SURVEY2
for i = 0 to 414
  begin
  forall rules(i, compl > 80 or (compl > 60 and supp > 50))
     print "decision =", i, "  class = ", class, "  rno = ", rno
  end
end
```

Figure 7. A KGL-1 script for the PARENT-CHILD database exploration scout.

Figure 8 shows an example of a strong rule generated by the knowledge scout, which describes a subset of the 1618 cases (*P*) in which the parent said that the child does not get into trouble with the law. There were 65 cases in which the parent said the child does get into trouble with the law (negative cases).

Cases in which parent said that child that does not get into trouble with the law are characterized by:

		p	*n*	*compl*	*cons*
[Child's frequency of lying about age ≤ *1 per week*]		1512	52	93%	96%
[Child says stealing over $50 is *wrong* or *very wrong*]		1587	60	99%	96%
[Child hired prostitute in past year ≤ *twice*]	1603	63	99%	96%	
[Child stole car in past year ≤ *twice*]	1612	64	99%	96%	
Rule Total:		*1376*	*54*	*85%*	*96%*

Figure 8. An example of an attributional rule generated in Study 2.

As shown in Figure 8, the found rule covers 1376 positive cases and 54 negative cases, and thus can be viewed as a strong pattern. In the rule, all conditions had completeness of 99% and consistency 96%, except for the first condition that had completeness 93%.

The study have shown that the proposed approach produces rules easy to interpret and understand. While the goal of the study was not to produce new sociological knowledge, the obtained results show that the proposed approach has a potential for producing such knowledge.

7 Summary and Future Research

This paper presented a novel methodology for integrating machine learning and inference methods with database operators for the purpose of conducting complex data mining and knowledge discovery operations. We introduced the concept of a knowledge scout as a software agent that utilizes resources of an inductive database to search for and synthesize target knowledge, that is knowledge of interest to a particular user or group of users. Such knowledge may include, e.g., strong patterns, relationships among different groups of attributes, hypotheses about future datapoints, qualitative data descriptions, plausible knowledge derived from generated hypotheses, etc.

An inductive database was defined as a database system that integrates conventional database with inductive inference and other data mining operators through a single knowledge generation language. Such a language includes conventional database operators with operators for conducting inductive inference and managing knowledge in the knowledge base. Knowledge scouts are defined by scripts in the knowledge generation language. An initial version of such a language, KGL-1, implemented in the INLEN-3 inductive database system, has been briefly described.

An important aspect of this research is that knowledge generated by knowledge scouts is expressed in a form that is easy to interpret and understand. This feature is due to the employment of attributional calculus, a highly expressive but computationally simple description language. Association graphs were introduced as a means for visually and abstractly representing multi-argument relationship defined by attributional rules.

Among topics for future research is how knowledge scouts can build, update and employ models of the users' interests in order to synthesize knowledge that is most relevant for the user at a given stage of data exploration. Research on incremental learning and concept drift in machine learning appears to be highly relevant for such a task (e.g., Maloof and Michalski, 1999).

Another research topic is to scale up the KGL-1 inductive inference operators so that they can work efficiently with very large databases. Since there are many operators involved, such a research project will require a significant effort. An interesting topic is to integrate one of the existing knowledge base systems, for example, PARKA (Taylor, Stoffel, and Hendler, 1997; Stoffel,Taylor, and Hendler, 1997) with INLEN 3. Other topics for future research include the development of operators for temporal trend prediction and for scaling up the conceptual clustering operator.

.

Acknowledgments

The authors thank Jim Logan for providing the American Cancer Society database and discussing experiments done in Study 1. This research was conducted in the Machine Learning and Inference Laboratory at George Mason University under partial support from the National Science Foundation under Grants No. IIS-0012121, IIS-9904078 and IRI-9510644.

References

Agrawal, R., Imielinski, T. and Swami, A., Mining Association Rules between Sets of Items in Large databases, *Proceedings of the ACM SIG-MOD Conference on Management of Data*, 207-216, Washington, D.C., 1993.

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

Elliott, D., "National Youth Survey (United States), Wave I, 1976," [Computer file]. ICPSR version, University of Colorado Behavioral Research Institute, Boulder, CO, 1977, distributed by Inter-University Consortium for Political and Social Research, Ann Arbor, MI, 1994.

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.

Fischthal, S., "A Description and User's Guide for CLUSTER/2C++ A Program for Conjunctive Conceptual Clustering," *Reports of the Machine Learning and Inference Laboratory*, MLI 97-10, George Mason University, Fairfax, VA, 1997.

Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Rajan, A., Stefanovic, N., Xia, B. and Zaiane, O.R., "DBMiner: A System for Mining Knowledge in Large Relational Databases," *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, p. 250-255, Portland, OR, 1996.

Imielinski, T., Virmani, A. and Abdulghani, A., "DataMine: Application Programming Interface and Query Language for Database Mining," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 256-261, 1996.

Kaufman, K. and Michalski, R.S., "Discovery Planning: Multistrategy Learning in Data Mining," *Proceedings of the Fourth International Workshop on Multistrategy Learning*, Desenzano del Garda, Italy, pp. 14-20, 1998.

Kaufman, K. and Michalski, R.S., "Learning From Inconsistent and Noisy Data: The AQ18 Approach," *Proceedings of the Eleventh International Symposium on Methodologies of Intelligent Systems*, Warsaw, June 1999 (to appear).

Maloof, M. and Michalski, R.S., "Selecting Examples for Partial Memory Learning," *Machine Learning* , Vol. 30, pp. 1-22, 1998.

Michalski, R. S., "Synthesis of Optimal and Quasi-Optimal Variable-Valued Logic Formulas," *Proceedings of the 1975 International Symposium on Multiple-Valued Logic*, Bloomington, Indiana, pp. 76-87, 1975.

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, Palo Alto: Tioga Publishing*, pp. 83-129, 1983.

Michalski, R.S., "Seeking Knowledge in the Deluge of Facts," *Fundamenta Informaticae*, Vol. 30, pp. 283-297, 1997.

Michalski, R.S., "NATURAL INDUCTION: Theory, Methodology and Applications to Machine Learning and Knowledge Mining," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, to appear.

Michalski R. S. and Kaufman, K., "Data Mining and Knowledge Discovery: A Review of Issues and Multistrategy Methodology," in Michalski, R.S., Bratko, I. and Kubat, M. (eds.), *Machine Learning and Data Mining: Methods and Applications*, London, John Wiley & Sons pp. 71-112, 1998.

Michalski, R.S. and Stepp, R., "Automated Construction of Classifications: Conceptual Clustering versus Numerical Taxonomy," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 4, pp. 396-410, July, l983.

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.

Taylor M., Stoffel, K., and Hendler, J., Efficient Management of Very Large Ontologies, *Proceedings of the AAAI Conference*, 1997.

Stoffel K., Taylor M., and Hendler, J., "Ontology-based Induction of High Level Classification Rules," *Proceedings of the SIGMOD Datamining and Knowledge Discovery Workshop*, May 1997.

Zhang, Q. and Michalski, R.S., "KV: A Knowledge Visualization System Employing General Logic Diagrams," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, 1999 (to appear).