Symbolic methodology for numeric data mining

Boris Kovalerchuk

Engenii Vityaev

Symbolic Methodology for Numeric Data Mining

Boris Kovalerchuk[a*], Evgenii Vityaev[b]
[a] *Central Washington University, Ellensburg, WA 98926-7520, USA,*
E-mail: borisk@cwu.edu
[b] *Institute of Mathematics, Russian Academy of Science, Novosibirsk, Russia, 630090.*
E-mail:evityaev@yahoo.com

Abstract. Currently statistical and artificial neural network methods dominate in data mining applications. Alternative relational (symbolic) data mining methods have shown their effectiveness in robotics, drug design, and other areas. Neural networks and decision tree methods have serious limitations in capturing relations that may have a variety of forms. Learning systems based on symbolic first-order logic (FOL) representations capture relations naturally. The learned regularities are understandable directly in domain terms that help to build a domain theory. This paper describes relational data mining methodology and develops it further for numeric data such as financial and spatial data. This includes (1) comparing the attribute-value representation with the relational representation, (2) defining a new concept of *joint relational representations,* (3) a process of their use, and Discovery algorithm. This methodology handles uniformly the numerical and interval forecasting tasks as well as classification tasks. It is shown that Relational Data Mining (RDM) can handle multiple constrains, initial rules and background knowledge very naturally to reduce the search space in contrast with attribute-based data mining. Theoretical concepts are illustrated with examples from financial and image processing domains.

Keywords: Data mining, relational data mining, numerical data, first order logic, probabilistic first order logic rules, stock market, image processing, edge detection.

1. Introduction

Historically, methods based on *attribute-value languages (AVLs)* have been most popular in applications of learning algorithms. One of the reasons is that in many areas training data are naturally described by attributes of individual entities such as stock prices or pixel intensities. Sometimes the actual values of X and Y are not available or not accurate and relation Price(X) > Price (Y) or Intensity(X) > Intensity(Y) is the only data available. Well-known, relatively simple and efficient *neural networks* and *decision trees* methods are typical examples of methods based on AVLs. However, these methods have serious limitations in **capturing relations**. Learning systems based on symbolic **first-order logic (FOL)** representations capture relations naturally. These methods have been successfully applied to many problems in chemistry, physics, medicine and other fields [4-6, 10-11, 18, 34, 39-41]. It was stated in these publications that the results obtained with relational methods using real industrial or environmental data are better than with any other known approach, with or without machine learning. Such tasks as mesh

[*] Corresponding author.

design, mutagenicity, and river water quality exemplifies successful applications. Domain specialists appreciate that the learned regularities are **understandable** directly in domain terms. Financial and spatial applications can specifically benefit from these methods. It is noted in [17] that lack of comprehension causes concern about the credibility of the result when neural networks are applied to *risky domains*, such as patient care and financial investment.

Traditionally, FOL methods were pure *deterministic techniques*, which originated in logic programming. There are well-known problems with deterministic methods—handling data with a significant level of *noise***.** This is especially important for financial and spatial data, which typically have a very high level of noise. To utilize advantages of human-readable forecasting rules produced in relational data mining, *logical relations (predicates)* should be developed for financial and spatial problems. These predicates should be **interpretable** in ordinary financial and spatial terms like stock prices, interest rates, trading days, pixel RGB colors, intensities, and so on. In this way, relational methods can produce valuable understandable rules in addition to the forecast.

Using this technique a specialist can evaluate the *performance of the forecast and discovery* as well as a *forecasting rule*. Hybridizing the pure logical relational data mining methods with a probabilistic approach ("probabilistic laws") has many advantages. Such hybrid methods use probabilities over logical formulas [7, 14, 21, 24, 29, 30, 36, 54, 56, 57]. The Discovery method is one of the few Hybrid Probabilistic Relational Data Mining methods developed and applied to financial data.

Below we outline the FOL approach and describe a new *hybrid relational and probabilistic technique* that handles **numerical data** efficiently. Traditional data mining methods focus on classification tasks. Relational data mining conceptually can handle (1) *classification*, (2) *interval* and (3) *numerical forecasting* tasks with noise uniformly. We advocate *relational learning mechanisms*, which combine advantages of rule induction, analytical learning and statistical paradigms such as **statistical significance, explanatory power** and a **highly-expressive language**. Specifically rule induction and analytical methods have strong capabilities for explaining discovered patterns, statistical methods ensure reliability of these patterns and the analytical methods use a highly expressive language (first-order predicate language) to ensure that complex patterns will not be overlooked.

The emphasis of our study is on development of numerical relational methods including relational representation of numeric data. How can one move from a real numerical measurement to a first-order logic representation? This is a non-trivial task [32]. For example, how does one represent temperature measurement in terms of first-order logic without losing the essence of the attribute (temperature in this case) and without inputting unnecessary conventional properties? For instance, Fahrenheit and Celsius zeros of temperature are arbitrary conventions in contrast with the Kelvin scale where zero is the lowest possible temperature (the physical zero). Therefore, incorporating properties of the Fahrenheit zero into first-order rules may force us to discover/learn properties of this convention along with more significant scale invariant forecasting rules. Learning algorithms in the space with those kinds of arbitrary properties may be very time consuming and may produce inappropriate rules.

We use the relational data mining (RDM) term, RDM, in parallel with the earlier terms *Inductive Logic Programming (ILP)* and *First Order Logic (FOL)* methods to emphasize the goal -- discovering **relations**. The terms ILP and FOL reflect the technique for discovering relations -- logic programming and FOL. In particular, discovering relational regularities can be done *without logical inference* and in languages of higher order. Therefore, we define **Relational Data Mining** as

> Discovering hidden relations (general logic relations) in numerical and symbolic data using background knowledge.

FOL systems have a mechanism to represent background financial and spatial knowledge in **human-readable and understandable form**. Obviously, understandable rules have advantages over forecast and discovery without explanations. One of the major obstacles to more effective use of the FOL methods is their limited facility for handling numerical data. This is one of the active topics of modern RDM research [4].

There are two types of numerical data in data mining:

- the numerical *target variable* and
- numerical *attributes* used to describe objects and discover patterns.

Traditionally FOL methods solve only classification tasks without direct operations on numerical data. The Discovery method handles an interval forecast of numeric variables with continuous values like prices along with solving classification tasks. In addition, Discovery handles **numerical time series** using the first-order logic technique, which is not typical for ILP and FOL applications.

Statistical significance is another challenge for deterministic methods. Statistically significant rules have an advantage in comparison with rules tested only for their performance on training and test data [33]. Training and testing data can be too limited and/or not representative. If rules rely only on them then there are more chances that these rules will not deliver a correct forecast on other data. This is a difficult problem for any data mining method and especially for deterministic methods including deterministic ILP. We address this problem in Section 6. Intensive studies are being conducted for incorporating a probabilistic mechanism into ILP [18, 38].

Knowledge Representation is an important and informal initial step in relational data mining. In attribute-based methods, the attribute form of data actually dictates the form of knowledge representation. Relational data mining has more options for knowledge representation. For example, attribute-based information should be transformed into the first order logic form. This knowledge includes much more than only values of attributes. There are many ways to represent knowledge in the first order logic language. One of them can *skip important information*; another one can *hide* it. Therefore, data mining algorithms may work too long to "dig" relevant information or even may produce inappropriate rules. Introducing *data types* [15] and concepts of *representative measurement theory* [31, 47] into the knowledge representation process helps to address this representation problem. In fact the measurement theory developed a wide set of data types, which cover data types used in [15].

It is well known that the general problem of rule generating and testing is *NP-complete* [22]. Therefore, the discussion above is closely related to the following questions. What determines the number of rules? When do we stop generating rules? The *number of hypotheses* is another important parameter. It has already been mentioned that RDM with first order rules allows one to express naturally a large variety of general hypotheses, not only the relation between pairs of attributes. These more general rules can be used for classification problems as well as for an interval forecast of a continuous variable. RDM algorithms face exponential growth in the number of combinations of predicates to be tested. A mechanism to decrease this set of combinations is needed. Section 6 addresses these issues using a data type system and the representative measurement theory approach. Type systems and measurement theory approaches provide better ways to generate only *meaningful hypotheses* using syntactic information. A probabilistic approach also naturally addresses knowledge discovery in situations with **incomplete or incorrect domain** knowledge. In this way the individual training cases are not generalized beyond the limits of statistically significant rules.

2. Examples

Examples in this section illustrate (1) attribute-value object representation vs. relational representation, (2) first order logic rules with one and two arguments, (3) the difference between IF-Then first-order logic rules and more traditional IF-Then propositional logic rules, (4) multi-attribute object representation vs. *joint relational representation*, and (5) steps of data mining based on joint relational data representation. These examples are derived from financial and image processing domains.

2.1. Example 1: attribute-value object representation vs. relational representation for data mining

Table 1 illustrates an attribute-value object representation. The first two lines represent some objects from a training data set. The last line represents an object without a value for the target attribute. The target value needs to be predicted for this object, i.e., stock price for the next day, 01.05.99. Each attribute-value pair can also be written as a name of an attribute and its value.

Table 1. Attribute-value object presentation

Attribute: date	*Attribute 1: Stock price on date t*	*Attribute 2: Volume (number of shares) traded on date t*	*Attribute 3: Target-- stock price on date t+1*
Value: 01.02.99	Value: $ 60.6	Value: 1,000,000	$53.8
Value: 01.02.99	Value: $ 53.8	Value: 700,000	$54.6
Value: 01.03.99	Value: $ 54.6	Value: 800,000	$56.3
Value: 01.04.99	Value: $ 56.3	Value: 840,000	

For instance, the following **rule 1** can be extracted from table 1:

> *IF stock price today is more than $60 AND trade volume today is greater than 900,000*
> *THEN tomorrow stock will go down.*

This rule can be written more formally:

> *IF StockPrice(t)>$60 AND StockTradeVolume(t)> 900,000*

THEN Greater(StockPrice(t+1), StockPrice(t))

Rule 2 is also true for table 1:

IF stock price today is greater than stock price yesterday AND
trade volume today is greater than yesterday THEN tomorrow stock price will go up.

Rule 2 also can be written more formally:

IF Greater(StockPrice(t), StockPrice(t-1)) AND
Greater(StockTradeVolume(t),StockTradeVolume(t-1)) THEN Stock-Price(t+1)>StockPrice(t)

Note, actually rule 2 is true for table 2 because table 1 does not have examples that contradict this rule. However, table 1 has only one example (t=01.03.99) confirming this rule. Obviously, table 1 is too small to derive reliable rules. Table 1 and presented rules are used just for illustrating that *attribute-value methods were not designed to discover rules 1 and 2* from table 1 directly**.** Both rules involve *relations between two objects* (records for two trading days t and (t+1)):

StockPrice(t+1)>StockPrice(t) and
Greater(StockTradeVolume(t),StockTradeVolume(t-1)).

Special **preprocessing** is needed to *create additional attributes*, such as

StockUp(t) = Greater(StockPrice(t), StockPrice(t-1)).

In logic terms attributes StockUp(t) and VolumeUp(t) are *monadic (unary) predicates* (Boolean functions with only one argument).Adding relations like

Greater(StockTradeVolume(t),StockTradeVolume(t-i))

with 2, 3, ...i days ahead to the set of attributes creates a huge set of relations.

The first order language differs from a propositional logic language mainly by the presence of **variables**. Therefore, a language of monadic functions and predicates is a first order logic language, but a very restricted language. A language of monadic functions and predicates was not designed to represent relations that involve *two, three or more objects*. The domain (background) knowledge that can be used in the learning process of attribute-value methods is of a *very restricted form*. Moreover, other relations from a database cannot be used in the learning process if they are not incorporated into a single attribute-value table [11].

2.2. Example 2: first-order rules vs. propositional rules

There is a lot of **confusion** about the difference between logical attribute-value methods and relational methods. At first glance, they do the same thing -- produce "IF-Then" rules and use logical expressions. Dzeroski [11] presented a fragment of a relational database for the potential customers of an enterprise to illustrate the difference. Table 2 presents a similar fragment of a relational database for corporate credit card holders. We wish to discover patterns in this table useful for distinguishing potential new cardholders from those who are not.

An attribute-value learning system may use Age, Sex and the Number of supervised associates from table 2. In this way, the following two patterns could be discovered with **monadic functions** Num_of_Supervised(Person) and Potential-Cardholder(Person):

Rule 1: *IF Num_of_Supervised(Person) ≥ 100*
THEN Corporate_Cardholder(Person)
Rule 2: *IF Sex(Person)=F AND Age(Person) ≥ 38*
THEN Corporate_Cardholder(Person)

Using a first order language with a two-argument predicate Colleague-of(person, colleague) the following pattern can be found:

Rule 3: *IF Colleague-Of(Person, Colleague)*
AND Corporate_Cardholder(Person) THEN Corporate_Cardholder(Colleague).

Table 2. Database Relation "Potential-Corporate-Credit-Cardholder" (Attribute-value table)

Person	*Age*	*Sex*	*Number of supervised associates*	*Corporate cardholder*
Diana Right	39	F	10	Yes
Carol Peterson	49	F	1000	Yes
Barbara Walker	24	F	20	No
Cindy Peck	47	F	20	Yes
Peter Cooper	35	M	100	Yes
Stephen Baker	54	M	200	Yes

Table 3. Database relation "Colleague-of" (Attribute-value table)

Person (CEO)	*Colleague (CFO)*
Peter Cooper	Diana Right
Stephen Baker	Cindy Peck

The last rule is much more meaningful than the first two formal rules. Rules 1 and 2 are discovered in an isolated file (table 2), but rule 3 is discovered using *two files simultaneously*. Table 2 represents a *single relation* in relational database terms. Table 3 represents another single relation. To find regularity involving records from *both tables* we need to use more expressive *first-order language*. Mathematically, first order languages generate such relations with *two, three and more variables***.** This explains the origin of another name used for relational data mining – **multi-relational data mining**.

2.3. Example 3: multiattribute object representation vs. joint relational representation

The example below illustrates multi attribute-value object representation vs. **joint spatial/spectral/temporal relational representation** in spatial domain. The numeric data representation carries relations between pixels **implicitly**, but relational representation carries them **explicitly**. Traditional numeric representation of imagery (raster representation) can be more compact than relational representation. Therefore, relations are not stored but computed as needed from imagery raster data. There is also an opposite effect when the "native" data representation is relational (see Table 3). In this case, conversion from relations to multi-attribute form can generate a much larger representation of the same data.

Implicit representation of relations has its *fundamental drawback.* The implicit relations that occur in imagery data and relations used by a particular KDD&DM method can **mismatch.** It can be difficult to see mismatch for relations presented only implicitly. As a result, data mining discoveries and forecasts may have no interpretation in the domain. Specifically neural network based stock market forecast has no explanation beyond a possible good interpolation of the historic stock data.

The fundamental challenge is invention of both meaningful and useful relations from noisy numeric data because not every meaningful relation is useful and not every useful relation is meaningful.

In image processing and spatial domain typical data are 2-D images, 3-D scenes, video (a set of n 2-D frames), and multidimensional cube of multispectral data (a set of n 2-D images). These data have both spectral and spatial relations of the pixels. The link between these relations should not be lost in modeling. These relations should not be separated to independent spectral and spatial components. Relations below carry such integrity.

Let p=(x,y,s) be a pixel at the location (x,y) with a spectral characteristic s=s(p), where s is scalar for panchromatic images (pixel intensity) and a vector for multispectral images. Predicate PC(I) denotes that image I is a panchromatic, predicate CO(I) means that I is color image, and MS(I) means that I is multispectral cube. We define several "Between" and "Edge" relations for any three pixels, p_1, p_2, and p_3 in the image I that are located in the same straight line. See Table 4. The relations that we defined in Tables 4 and 5 are relative spectral/spatial/temporal relations that are invariant to measurement units and distances.

The experimental base for these relations is eye gaze motion lines recorded in many experiments. An experiment can be designed to discover experimentally actual joint relations captured by a human eye. There are also opportunities to base joint relations on retina modeling research and measurements [19]. Specifically it can help to set up axioms (that can cut search time) for edge detection or invariance property of the joint relation due to impact of shadows, shading and highlights transforms. If these impacts are monotone then they do not change many relations we defined in tables 4 and 5. Otherwise non-monotone shading effects applied to our relations can help to find image areas modified by shading that can be used for edge detection.

Table 4. Joint spectral/spatial relations for individual images

Relation	*Interpretation*
LB(p_1,p_2,p_3) = Location_Between (p_1,p_2,p_3)	p_2 is located between pixels p_1 and p_3 on a straight line

NPB(p_1,p_3)=NoPixelsBetween(p_1,p_3)	No pixels in between p_1 and p_3.
SB((p_1,p_2,p_3) = SpectrBetween(p_1,p_2,p_3)	Spectral characteristic of p_2 is between spectral characteristic of p_1 and p_3 (no limit on mutual location of p_1,p_2,p_3)
SH((p_1,p_2,p_3) = SpectrHigher(p_1,p_2,p_3)	Spectral characteristic of p_2 is higher than spectral characteristic of p_1 and p_3 (no limit on mutual location of p_1,p_2,p_3)
SL((p_1,p_2,p_3) = SpectrLower(p_1,p_2,p_3)	Spectral characteristic of p_2 is lower than spectral characteristic of p_1 and p_3 (no limit on mutual location of p_1,p_2,p_3)
LBSB(p_1,p_2,p_3) = LB(p_1,p_2,p_3) & SB(p_1,p_2,p_3)	Both LB(p_1,p_2,p_3) & SB(p_1,p_2,p_3) are true, that is p_2 is between p_1 and p_2 in location and spectral characteristic.
LBSH(p_1,p_2,p_3) = LB(p_1,p_2,p_3) & SH(p_1,p_2,p_3)	Both LB(p_1,p_2,p_3) & SH(p_1,p_2,p_3) are true, that is p_2 is between p_1 and p_2 in location and higher than both p_1 and p_3 in spectral characteristic.
LBSL(p_1,p_2,p_3) = LB(p_1,p_2,p_3) & SL(p_1,p_2,p_3)	Both LB(p_1,p_2,p_3) & SL(p_1,p_2,p_3) are true, that is p_2 is between p_1 and p_2 in location and lower than both p_1 and p_3 in spectral characteristic.
E(p)	User assigned predicate, p is on the edge, "*ground truth*". It is reasonable to assume that a user identifies edges using joint spectral/spatial relations consciously or unconsciously.
Edge(p_1,p_2,p_3) = LB(p_1,p_2,p_3)&E(p_2)	p_2 is on the edge between p_1 and p_3
ET(p) $\Leftrightarrow$s(p)>T	Threshold based edge, spectral value s(p) of pixel p is grater than a threshold T.
ETLB((p_1,p_2,p_3) = ET(p_2) & LB(p_1,p_2,p_3)	Threshold and location based edge, spectral value s(p) of pixel p is grater than a threshold T and p_2 is located between p_1 and p_3.

Relations LBSB(p_1,p_2,p_3), LBSH(p_1,p_2,p_3), LBSL(p_1,p_2,p_3) are **joint spectral/spatial relations**, but ET(p) is only *spectral relation*. We can compute its value without knowing its location. Similarly, LB(p_1,p_2,p_3) is only spatial relation. We can compute its value without knowing spectral characteristics of pixels p_1,p_2 and p_3. For short, we will call a joint spectral/spatial relation a **spatiospectral relation**. For **3-D scenes** all relations are modified – each pixel p is substituted by 3D volume element (**voxels**) v in Table 4. Similarly, for **multispectral imagery** (cube) each pixel p is substituted by multidimensional element m =(x,y,s_1,s_2,…,s_n), where (x,y) is a location of the element and s_i is a spectral value for band m_i.

Table 5 presents the second level of joint relations, i.e., relations on sets of pixels not only individual pixels. These relations can be used to learn and discover edges in different parts of the image I. For **dynamic 3-D scenes** all relations are modified – each pixel p is substituted by 3D volume element (voxels) v in Table 6. Similarly, for **multispectral imagery** (cube) each pixel p is substituted by a multidimensional element m = (x,y,s_1,s_2,…,s_n), where (x,y) is a location of the element and s_i is a spectral value for band m_i. Alternatively, for **multispectral imagery** time t is substituted by a spectral band d in CSSE(p_1,p_2,p_3,d_1,d_2).

Table 5. Spatiospectral relations of the second level for an individual image

Relation	*Interpretation*
Edge(D,S) $\Leftrightarrow \forall$ $p_2 \in$ D, $\exists$ p_1, $p_3 \in$ S Edge(p_1, p_2, p_3)	D is an edge (a set of edge pixels) for a set of pixels S based on all available information including human edge detection skills.
SSE(D,S) = SpatioSpectralEdge(D, S) $\Leftrightarrow$ $\forall$ $p_2 \in$ D, $\exists$ p_1, $p_3 \in$ S LBSH(p_1,p_2,p_3)	D is an edge for a set of pixels S based only on spatio-spectral knowledge. It can differ from Edge(D.S).
Edge(D,I)	Edge(D,I) identifies all edges D in image I.
SSE(D,I) = SpatioSpectralEdge(D, I)	SSE(D,I) identifies all spatiospectral edges D in image I.

Table 6. Spectral/spatial/temporal relations for motion imagery

Relation	*Interpretation*
CSSE(p_1,p_2,p_3,t_1,t_2)= ChangeSpatioSpectralEdge(p_1,p_2,p_3,t_1,t_2) $\Leftrightarrow$ (LBSH(p_1,p_2,p_3,t_1) $\neq$ LBSH(p_1,p_2,p_3,t_2)	Change of spatiospectral edge relation from time t_1 to time t_2 for pixels p_1,p_2 and p_3.
CSSE(D, I, t_1,t_2)) = ChangeSpatioSpectralEdge(D, I,t_1,t_2) $\Leftrightarrow$ $\forall$ $p_2 \in$ D $\exists$ p_1,$p_3 \in$ I CSSE(p_1,p_2,p_3,t_1,t_2)	SSE(D,I) identifies all changed edges D in image I between t_1, and t_2.

In summary, the data type for still images, static 3D scenes and multispectral imagery includes relations defined above:

RDT=RelationalDataType = $\langle$LB, SB, SH, SL, LBSB, LBSH, LBSL, E, Edge, ET, ETLB$\rangle$.

These relations are augmented by relations CSSE and LBSH for motion imagery, dynamic 3D scenes and multispectral imagery. The relational data mining approach can use any relations listed in Tables 4-6. Many other relations can be generated from imagery, but joint spectral/spatial/temporal relations have important advantages. They are at the low level and can be easily computed. In contrast, a high level spatial only relations such as "at", "nearby", "in the vicinity", and "far away" are difficult to interpret exactly and compute.

The relations based solely on qualities such as distance, size, volume, spatial order and time have no direct reference to spectral relations. Pure spatial relations abstract and separate spatial relations from pixel intensities in images and scenes. These abstraction skills are learned in the early childhood and are critical for human conscious visual and analytical reasoning. In this way, human natural **unconscious abilities to percept joint spectral/spatial relations** are augmented. But the computer vision systems has nothing to augment, it fundamentally lacks human natural capabilities to process joint relations.

In Table 4, relations LBSB, LBSH and LBSL are proposed to reconstruct some joint spatiospectral relations. In addition, these relations with three arguments capture properties of the images deeper than relations between only two pixels or properties of individual pixels. The next step to capture even deeper properties of the images is to generate joint multilevel spectral/spatial/temporal relations (data types).

The importance of joint relations has been recognized in the multispectral research and classic mathematical morphology theory was used as a tool [59], however it was stated in [59] that most available attempts are based on the consideration of spectral information separately from spatial information, and thus the two types of information are not treated simultaneously. A relatively common approach to integration is computing **spatiospectral frequencies** and distributions. This approach generalizes spectral values relative to their locations, but does not capture individual relations *explicitly*. We model actual joint relations not generalized frequencies and use *probabilities on relations* not raw data to model the effect of noise and other factors.

2.4. Steps of data mining based on joint relational data representation for edge detection

Having training data (images) relational data mining approach can learn an algorithm of how to detect edges. Ideally, the learning should produce an algorithm to classify pixels as clear edges, noisy edges and no edges. The advantage of such *automatic learning approach* is that it avoids manual design of an edge detection algorithm. This is its fundamental advantage over modern manual design of edge detections algorithms. It discovers an algorithm automatically instead of manually designing it. Potentially an automatic approach based on joint spectral/spatial relations can rediscover known edge detection algorithms and mimic human edge detection process. The relational learning has an advantage over edge learning based on an assemble of the neural networks [3] because of abilities to interpret the discovery and check its validity conceptually beyond acceptable accuracy on specific test images that are always limited.

The process of automatic relational discovery of an edge detection algorithm can be as follows:

Step 1: Identify training and testing data (images, $\{I_{training}\}$, $\{I_{testing}\}$) and their data types (relations on data, RDT)

Step 2. Identify the target relation/predicate from RDT. For the edge detection problem, it is E(p), that is human detected edge pixels “ground truth” (see Table 4).

Step 2: Learn E(p) using some relations $\mathbf{P}=\{P_i\}$ from RDT, that is discover a rule/algorithm R_{edge}

$$R_{edge}: \; P_1 \& P_2 \& \ldots P_k \Rightarrow E(p)$$

The learning starts from testing shortest hypotheses that contain only individual relations P_i

$$P_i \Rightarrow E(p)$$

More relations are added from set **P** to the if-part of the rule if the hypothesis failed or cannot be improved on training and testing images in the statistical test. Search for a rule is shortened by eliminating some input data to be tested. It is done by using RDT axioms such as:

1) $\forall p_1,p_2,p_3$ LBSB(p_1,p_2,p_3) = LBSB(p_3,p_2,p_1)
2) $\forall$ p_2 E(p_2)&PC(I) $\Rightarrow$ $\exists p_1,p_2$ SH((p_1,p_2,p_3)

We do not need to test LBSB(p_3,p_2,p_1) in the rule if for LBSB(p_1,p_2,p_3) rule was already satisfied. The first axiom is a commutativity axiom for p_1 and p_3. The second one states that if p is on the edge in a panchromatic image I then pixels p_1 and p_3 should exist on a line with p_2 such that their intensity is less then the intensity of p_2. Note that in the axiom 2 we cannot reverse the implication because SH is only necessary but insufficient condition to be able to identify edge pixels.

Table 7 presents some hypotheses of a rule (an algorithm) for edge detection. These hypotheses can be tested on images by using relational data mining methods.

Table 7. Hypotheses for rules/algorithms

Rule	*Interpretation*
$\forall$ p_1,p_2,p_3 LBSH(p_1,p_2,p_3) $\Rightarrow$ E(p_2)	Spatiospectral edges LBSH are fully consistent with human assigned edges (“ground truth”). If this hypotheses in confirmed in images $\{I_{training}\}$ and $\{I_{testing}\}$ then it gives an algorithm to detect edges.
$\forall$ p_1,p_2,p_3, p_4,p_5 LBSH (p_1,p_2,p_3) & LBSH (p_2,p_3,p_4) & LBSH (p_3,p_4,p_5) $\Rightarrow$ E(p_2) & E(p_4)	If pixels p_1,p_2,p_3, p_4 and p_5 all are on a the straight line between p_1 and p_5 then only p_2 and p_4 are edge pixels. Pixel p_3 is not edge pixel, because it is between two edge pixels. For instance, for a wide road, only pixels that identify shoulders of road are edge pixels, but internal road pixels are not edge pixels.
$\forall$ p_1,p_2,p_3 LBSH(p_1,p_2,p_3) & NPB((p_1,p_2) $\Rightarrow$ E(p_2).	If pixel p_2 is in the edge between p_1 and p_2 according to LBSH and no pixels in between p_1 and p_2 then p_2 is an edge pixel.

The further work in this area can be to restore spatial relations using spectral relations in a *jigsaw puzzle*, where global spatial relations are lost, but local are available and the goal is to restore global spatial relations using every local ones. The typical approach again is based on frequencies of the individual numerical value not relations. Relational approach has advantages over this one.

The next step of integration of spectral/spatial/temporal relations audio/text relations is adding relations of image audio and text annotations for an image, video or a multispectral data cube. This subject is closely related to the emerging multimedia relational data mining [52].

3. Relational data mining paradigm

As we discussed in the previous section, attribute-value languages are quite restrictive in inducing relations between different objects explicitly. Therefore, richer languages were proposed to express relations between objects and to operate with objects more complex than a single tuple of attributes. Lists, sets, graphs and composite types exemplify complex objects. These more expressive languages belong to the class of first order logic languages (see definitions in [49, 11, 34]). These languages support *variables, relations,* and *complex expressions*. FOL methods can discover regularities using several tables (relations) in a database, such as Tables 3 and 4, but the **propositional approach** requires creating a *single table,* called *a universal relation* (relation in the sense of relational databases) [11].

Relational data mining methods should be able to solve numerical and *interval forecasting tasks* along with classification tasks such as presented above. This requires modifying the concept of positive and negative training examples E+ and E- and modifying the concept of deriving (inferring) training examples from background knowledge and a predicate. A relational algorithm, called Discovery is able to solve numerical and interval forecasting tasks. This algorithm operates with a set of training examples E. Each example is amended with a target value like is done in Table 2, where attribute #3 is a target attribute--stock price for the next day. This is a numerical value. There is no need for Discovery to make this target discrete to get a classification task. Therefore, more generally, a **deterministic relational data mining (DRDM)** mechanism is designed for forecasting tasks, including classification, interval and numerical forecasting. Similar to the definition for classification tasks, for general deterministic RDM *background knowledge B* is expressed as:

- a set of predicate definitions,
- training examples E expanded with target values T (nominal or numeric), and
- set of **hypotheses** {Gk} expressed in terms of predicate definitions.

Using this background knowledge a RDM system will construct a set of predicate logic formulas $\{H_i\}$ such that: the target forecast for all the examples in E can be *logically derived* from B and the appropriate H_i .

Example 3. Let us consider Rule 2 discovered from Table 2.

IF Greater(StockPrice(t), StockPrice(t-1))
AND Greater(StockTradeVolume(t) StockTradeVolume(t-1))
THEN StockPrice(t+1)>StockPrice(t)

This rule represents logical formula H, and table 2 represents training examples E. These two sources allow us to derive the following relation logically for date (t+1)=(01.04.99):

$$StockPrice(01.04.99)>54.6 \qquad (1)$$

assuming that t=(01.03.99). This is consistent with actual StockPrice(01.04.99)=56.3 for date 01.03.99. Rule 1 from the same Example 1 in Section 4 represents logical formula H1, but this rule is not applicable to t=(01.04.99). In addition, other rules can be discovered from Table 2.

For instance,
IF StockPrice(t)<$60 AND StockTradeVolume(t) < $90000
THEN Greater($60,StockPrice(t+1))
This rule allows us to infer

$$StockPrice(01.04.99)< 60 \qquad (2)$$

Combining (1) and (2) we obtain

$$60>StockPrice(01.04.99)>54.6 \qquad (3)$$

With more data we can narrow the interval (54.6, 60) for t=(01.04.99). A similar logical inference mechanism can be applied for t=(01.04.99) to produce a forecast for (t+1)=(01.05.99). This example illustrates one of the ideas used in the hybrid Discovery method for numeric interval forecast.

In contrast with the deterministic approach, in **Hybrid Probabilistic Relational Data Mining** background knowledge B is expressed as:

- A set of predicate definitions,
- Training examples E expanded with target values (nominal or numeric), and
- A set of *probabilistic hypotheses* $\{G_k\}$ expressed in terms of predicate definitions.

Using this background knowledge, a system constructs a set of predicate logic formulas $\{H_i\}$ such that: *Any example in E is derived from B and the appropriate H_i probabilistically, i.e., statistically significantly.* Applying this approach to (3) *60> StockPrice(01.04.99)>54.6,* we may conclude that although this inequality is true and is derived from table 2 it is not a statistically significant conclusion. It may be a property of a training sample, which is too small. Therefore, it is risky to rely on statistically insignificant forecasting rules to derive this inequality.

4. Theory of RDM

4.1. Data types in relational data mining

A **data type** (type for short) in modern object-oriented programming (OOP) languages is a rich data structure, $\langle \mathbf{A,P,F} \rangle$. It consists of *elements* $\mathbf{A}=\{a_1,a_2,...a_n\}$, *relations* between elements $\mathbf{P}=\{P_1,P_2,...P_m\}$ and meaningful *operations* with elements $F=\{F_1,F_2,...,F_k\}$. Operations may include two, three or more elements, e.g., c = a # b, where # is an operation on elements a and b producing element c. This definition of data type formalizes the concept of a **single-level data type**. For instance, a single-level graph structure ("stock price" data type) can be created with nodes reflecting individual stock prices and edges reflecting relations between stock prices (<, =, >). These graph structures (values of the data type) can be produced for each trading day -- StPr(1), StPr(2),..., StPr(t) – generating a time series of graph structures.

A **multilevel data type** can be defined by considering each element a_i from **A** as a composite data structure (data type) instead of as an atom. To introduce a multilevel stock price data type, stocks are grouped into categories such as high-tech, banking and so on. Then relations (<, =, >) between the average prices of these groups are defined. Traditional attribute-value languages operate with much simpler single-level data types. Implicitly, each *attribute in attribute-value languages reflects a type*, which can take a number of possible values. These values are elements of **A**. For instance, attribute "date" has 365 (366) elements from 01.01.99 to 12.31.99. There are several meaningful relations and operations with dates: **<, =, >,** and middle(a,b). For instance, the operation middle(a,b) produces the middle date c=01.05.99 for inputs a=01.03.99 and b=01.07.99. It is common in attribute-value languages that a data type such as a date is given as an **implicit data type** (see example 4 below). Usually in AVLs, relations **P** and operations **F** are not expressed explicitly. However, such data types *can be embedded explicitly into* *attribute-value languages.*

Example 4. Let us consider data type "trading weekdays", where a set of elements A consists of {Mon, Tue, Wed, Thu, Fri}. We may code these days as {1,2,3,4,5} and introduce a distance ρ(a,b)=|a-b| between them using these numeric codes. For instance, ρ(Mon,Tue)=ñ(1,2)=|1-2|=1 and ρ(Fri,Mon) = ρ(5,1)=|5-1|=4. The last distance is natural if both Friday and Monday belong to *the same week*, but if Monday belongs to the next week it would be more reasonable to assign ρ(Fri,Mon)=1, because Monday is the next trading day after Friday. This is a property of **cyclical scales**. Different properties of cyclical scales are studied in representative measurement theory [31]. The "trading weekdays" data type is a **cyclical data type**. This distance has several properties, which are unusual for distances. For instance, it is possible that ρ(a,b) ≠ ρ(b,a),

Let us assume that weekday *a* always precedes weekday *b*. Under this assumption ρ(Fri,Mon) means a distance between current Friday and Monday next week, but ρ(Mon,Fri) means a distance between Mon and Fri during the same week. In this example, the requirement that a precedes b was not defined explicitly. In [26,27] we studied cyclical scales and suggested numeric and binary coding schemes preserving this property for a variety of cyclical scales.

A new **strongly typed programming language Escher** was developed to meet this challenge [15]. The Escher language is an important tool, which allows users to incorporate a variety of explicit data types developed in representative measurement theory into the programming environment. On the other hand, RDM can be successfully implemented using common languages like Pascal and C ++ [56-57].

4.2. Relational representation of examples

Relational representation of examples is the key to relational data mining. If examples are already given in this form, relational methods can be applied directly. For attribute-based examples, this is not the case. We need to *express attribute-based examples and their data types in relational form*. There are two major ways to express attribute-based examples using predicates: (1) *generate predicates* for each value and (2) use *projection functions* (see below). Table 8 presents an attribute-based data example for a stock.

Table 8. Attribute-based data example

Stock price, $	*Volume, x1000*	*Date*	*Weekday Stock*	*Event*
54.60	3067.54	01.04.99	Monday	New product

Generating predicates for each value. To express stock price $54.60 from Table 8 in predicate form, we may generate predicate P546(x), such that P546(x) = true if and only if the stock price is equal to $54.60. In this way, we would be forced to generate about 1000 predicates if prices are expressed from $1 to $100 with a $0.10 step. In this case, the ILP problem will be intractable. Moreover, the stock price data type has not yet been presented with the P546(x) predicate. Therefore, additional relations to express this data type should be introduced. For example, it can be a relation between predicates P546(x) and P478(x), expressing a property that stock price 54.6 is greater than 47.8. To avoid this problem and to constrain the hypothesis language for RDM, the **projection**

function was introduced [15]**.** This concept is described below.

Representation of background knowledge. ILP systems use two sorts of background knowledge: objects and relations between those objects. For example, objects are named by constants a,b,c and relations are expressed using these names, P(a,b)=true and P(c,b)=false. Use of constants is not very helpful because normally names do not carry properties of objects useful for faster data mining. In the approach suggested in [15], this is avoided. An object is "named" by the collection of all of its characteristics (*terms*).

For instance, term representation of stock information on 01.04.1999 can be written as follows:

> *StockDate(w)=01.04.1999 & StockPrice(w)=$54.60 &*
> *StockVolume(w)=3,067,540 & StockWeekday(w)=Mon &*
> *StockEvent(w)= "new product".*

Here *StockPrice* is a **projection function** which outputs stock prices (value of StockPrice attribute). Only naming of subterms is needed. This representation of objects (examples) is convenient for adding new information about an object (e.g., data types) and *localizing* information. For instance, subterm "StockEvent" permits one to localize such entities as reported profit, new products, competitor activity, and government activity.
In the example above the following **data types** are used:

- type weekday = {Mon, Tue, Wed, Thu, Fri},
- type price,
- type volume,
- type date,
- type event ={reported profit, new product, competitor's activity, government activity, ...},
- type stock = {price, volume, date, weekday, event}.

Type event brings a description of event related to the stock, e.g., published three month profit, new product, competitor's activity. This can be as a simple text file as a structured data type.
The representation of an example then becomes the **term** *Stock (54.6, 3067.54, 01.04.99, Mon, new product).* Notice that when using projection functions in addition to predicates it is possible, without the use of variables, to represent relational information such as the equality of the values of two attributes. E.g., projection function StockEvent together with the equality relation (=) are equivalent to predicate SameEvent(w,x): *SameEvent(w,x) StockEvent(x)=StockEvent(w).*

Thus, the distinction between different propositional and first-order learning tasks depends in part on the representation formalism.

Strongly typed languages. FOL systems use types to provide labels attached to logical variables. However, these are not the data type systems found in modern programming languages. All available literals in the Prolog language will be considered for inclusion if a naive refinement operator is used for Prolog [15]. These authors developed a new strongly typed ILP language, *Escher,* which employs a complex data type system and restricts the set of hypotheses by *ruling out many useless hypotheses.* The Discovery method (Section 4) employs another way to incorporate data types into data mining by adding a data type structure (relational system) into the background knowledge. Such a relational system is based on representative measurement theory.

Complex data types and selector functions. Each data type is associated with a relational system, which includes: (1) cardinality, (2) permissible operations with data type elements, and (3) permissible relations between data type elements.

In turn, each data type element may consist of its own subelements with their types. **Selector functions** [15] serve for extracting subterms from terms. Without selector functions, the internal structure of the type could not be accessed. Projection for selecting the i-th attribute requires the tuple type and a list of components (attributes) of the tuple. A list of components (attributes) requires the length of the list and the set of types of components.

The number of hypotheses. The most important feature of strongly typed languages is that they not only restrict possible values of variables, but also more importantly **constrain the hypothesis language.**

Table 9 summarizes information about data type features supported by different languages: ordinary attribute-based languages, attribute-based languages with types, first-order logic languages with types and ILP languages based on Prolog. This table is based on analysis from [15]. Strongly typed languages for numerical data are especially important for financial and spatial applications with prevailing numeric data.

Single Argument Constraints. Consider an example, the term stock(A,B,C,D,E) has a type definition of stock(price, volume, date, weekday, event). Having this type definition, testing rules with arguments such as (25.7, 90000, 01.04.99, 67.3, new product) is avoided because 67.3 does not belong to weekday type. Thus, this typing information is a useful simple form of background knowledge. Algorithms FOCL and Discovery take advantage of typing information. On the other hand, the well-known FOIL algorithm does not use type constraints to eliminate literals from consideration.

Table 9. Data types supported by data mining languages

Supported features of object representation	*Attribute-based language*	*Attribute-based language with types*	*First-order language with types*	*ILP based on Prolog language*
Formally expressed data type context	No	Yes	Yes	Yes
Attribute-value tuples	Yes	Yes	Yes	No
Explicitly induced relations between tuples	No	Yes	Yes	Yes
Data types of attributes expressed as in modern object-oriented languages	No	Yes	Yes	No
Mechanism to restrict the set of possible hypotheses using data types	No	Yes	Yes	No
Representing objects by terms using a projection function	No	Yes	Yes	No

Typing can be combined with **localized predicates** to *reduce the search* space. For instance, a localized relation *Greater_dates(A,B)* can be introduced to compare only dates with type information *Greater_dates(date,date)* instead of a universal relation *Greater(item, item)*. Similarly, a localized relation *Greater_$(A,B)*, type information *Greater_$(price, price)* can be introduced and applied for prices. This localized typing avoids the testing of some arguments (literals). For instance the localized predicate Greater_dates(A, B) should not be tested for literals of types such as *Greater_dates(stockprice, stockprice), Greater_dates(stockprice, date),* and *Greater_dates(date, stockprice).*

More generally, let $\{T_i\}$ be the types of already used arguments $\{x_i\}$ in predicate P. Predicate P should be tested for different sequences of arguments. If the type T_i of the already used i-th argument of P contradicts the type of a new i-th argument suggested for testing P, then the testing of the sequence which involves this new argument can be eliminated. This is a correct procedure only if a predicate is **completely localized**, i.e., only one type of argument is allowed for y_i. It is the case for the predicate *Greater_dates*, but it is not for the original predicate Greater defined for any items. This consideration shows that typing information *improves background knowledge* in two ways: (1) adding predicates and clauses about data types themselves and (2) refining and adding predicates and clauses about objects (examples). In such situations, typing can in the best case exponentially reduce the search space [15]**.** FOCL algorithm illustrates the benefit of typing. FOCL algorithm tested 3240 units and 242,982 tuples using typing as compared to 10,366 units and 820,030 tuples without typing. This task contained [44]:

- learning a predicate with six variables of different types and
- 641 randomly selected training examples (233 positive and 408 negative training examples).

Typing is very useful for data mining tasks with limited training data, because it can improve the *accuracy of the hypothesis* produced without enlarging the data set. However, this effect of typing is reduced as *the number of examples increases* [15, 44].

Existential variables. The following hypothesis illustrates existential variables:

IF (there exists stock w such that StockEvent(x)=StockEvent(w)) AND (Some other statement)
THEN StockPrice(x)>StockPrice(w)

The variable w is called *existential variables*. The *number of existential variables* like w and z provides one of the measurements of the *complexity of the learning task*. Usually the search for regularities with existential variables is a computational challenge.

4.3. First-order logic and rules

A **predicate** is defined as a binary function or a subset of a set $D=D_1\times D_2\times\ldots\times D_n$, where D_1 can be a set of stock prices at moment t=1 and D_2 can be stock price at moment t=2 and so on. Predicates can be defined **extensionally**, as a list of tuples for which the predicate is true, or **intensionally**, as a set of **(Horn) clauses** for computing whether the predicate is true. Let *stock(t)* be a stock price at t, and consider the predicate *UpDown(stock(t), stock(t+1), stock(t+2)),* which is true if stock goes up from date t to date t+1 and goes down from date t+1 to date t+2. This predicate is presented extensionally in the last column in Table 10.

Table 10. Stock data and UpDown predicate

Stock(t)	*Stock(t+1)*	*Stock(t+2)*	*Updown(, ,)*
$34	$38	$35	True
$38	$35	$35.50	False
$35.50	$36	$34	True
$36	$37	$38	False

It also can be presented **intensionally** using two other predicates Up and Down:

Up(stock(t),stock(t+1)) & Down(stock(t+1),stock(t+2))→ Up-Down(stock(t),stock(t+1),stock(t+2),

where Up(stock(t),stock(t+1)) ⇔ Stock(t+1) ≥ Stock(t), and
Down(stock(t),stock(t+1)) ⇔ Stock(t) ≤ Stock(t+1).

A literal is a predicate A or its negation (¬A)**.** The last one is called *a negative literal.* An unnegated predicate is called a *positive literal.* A *clause body* is a conjunction $A_1\&A_2\&...\&A_t$ of literals $A_1,A_2,...,A_t$. Often we will omit & operator and write $A_1\&A_2\&...\&A_t$ as $A_1A_2...A_t$. A **Horn clause** consists of two components: a *clause head* (A_0) and a *clause body* ($A_1A_2...A_t$). A clause head, A_0, is defined as a single predicate. A Horn clause is written in two equivalent forms: $A_0 \leftarrow A_1A_2...A_t$, *or* $A_1A_2...A_t. \rightarrow A_0$, where each A_i is a literal. The first form is common in applications and the second one is common in mathematical logic.

A collection of Horn clauses with the same head A_0 is called a **rule.** The collection can consist of a single Horn clause; therefore, a *single Horn clause* is also called a rule. Mathematically the term collection is equivalent to the OR operator (∨), therefore the rule with two bodies $A_1A_2...A_t$ and $B_1B_2...B_t$ can be written as $A_0 \leftarrow (A_1A_2...A_t \vee B_1B_2...B_t)$. A k-tuple, a functional expression, and a term are the next concepts used in relational approach. A finite sequence of k constants, denoted by $\langle a_1,...,a_k \rangle$ is called a **k-tuple** of constants. Constants are used as arguments in A_i and B_i. A function applied to k-tuples is called a **functional expression**. A **term** is a constant, variable or functional expression. Examples of terms are given in table 11. A k-tuple of terms can be constructed as a sequence of k terms. These concepts are used to define the concept of atom. An **atom** is a predicate symbol applied to a k-tuple of terms. For example, a predicate symbol P can be applied to 2-tuple of terms *(v,w),* producing an atom *P(v,w)* of arity 2. If P is predicate “>” (greater), *v=StockPrice(x)* and *w=StockPrice(y)* are two terms then they produce an atom: *StockPrice(x)* > *StockPrice(y),* that is, price of stock x is greater than price of stock y. Predicate P uses two terms v and w as its arguments. The number two is the arity of this predicate.

Table 11. Examples of terms

Expression	*Comment Term*	*Term ?*
x	Variable – stock x	Yes
MSFT	Constant (specific stock/index)	Yes
StockPrice(x)	Functional expression	Yes
TradeVolume(x)	Functional expression	Yes
Stock-Price(x)*TradeVolume(x)	Functional expression	Yes
Nasdaq(x)>StockPrice(x)	Incorrect	No
NASDAQ(x)	Predicate, literal (Stock x is traded on NASDAQ)	No
Stock-Price(x)>StockPrice(y)	Predicate(x,y), literal	No

If a predicate or function has k arguments, the number k is called **arity** of the predicate or function symbol. By convention, **function** and **predicate symbols** are denoted by Name/Arity. Functions may have variety of values, but predicates may have only Boolean values true and false. The meaning of the rule for a k-arity predicate is the set of k-tuples that satisfy the predicate. A tuple satisfies a rule if it satisfies one of the Horn clauses that define the rule. A *unary (monadic) predicate* is a predicate with arity 1. For example, NASDAQ(x) is unary predicate. **Predicates** defined by a collection of examples are called *extensionally defined predicates*, and predicates defined by a rule are called *intensionally defined predicates.* If predicates defined by rules then inference based on these predicates can be explained in terms of these rules. Similarly, the *extensionally defined predicates* correspond to the **observable facts** (or the *operational predicates*) [34]. A collection of intensionally defined predicates is also called *domain knowledge* or **domain theory.** Statements about a particular stock MSFT for a particular trading day can be written as:

StockPrice(MSFT) > 83, NASDAQ(MSFT), TradeVolume(MSFT)=24,229,000.

5. Background knowledge

5.1. Arguments constraints and skipping useless hypotheses

Background knowledge fulfills a variety of functions in the data mining process. One of the most important is reducing the number of hypotheses to be tested to speed up learning and make this process tractable. There are several approaches to reduce the size of the hypothesis space. Below two of them are presented. They use constraints on arguments of predicates from background knowledge B. The difference is that the first approach uses **constraints on a single argument** and the second one uses **constraints on several arguments** of a predicate defined in B. The first approach called **typing** approach is based on information about individual data types of arguments of a predicate. For instance, suppose only an integer can be the first argument of predicate P and only the date (M/D/Y) can be the second argument of this predicate. It would be wasteful to test hypotheses with the following typing P(date, integer), P(integer, integer) and P(date, integer). The only one correct type here is P(integer, date). The second approach is called **inter-argument constraints approach.** For example, predicate *Equal(x,x)* is always true if both arguments are the same. Similarly, it is possible that for some predicate P for all x P(x,x)=0. Therefore, testing hypotheses extended by adding Equal(x,x) or P(x,x) should be avoided and the **size of the hypothesis space explored can be reduced**. The value of inter-argument constraints is illustrated by the experimental fact that the FOCL algorithm, using **typing and inter-argument constraints**, was able to test 2.3 times less literals and examples than using only typing [44]. Table 12 summarizes properties of the two discussed approaches for reducing the number of hypotheses.

Table 12. Approaches for reducing hypothesis space

	Approach 1: Implementing a single argument constraint (typing)	*Approach 2: Implementing inter-argument constraints*
Definition	Properties of an individual argument of the predicate.	A relationship between different arguments of a predicate
Example of constraints	Only an integer can be the first argument of a predicate. Only date (M/D/Y) can be the second argument of the predicate.	All of the variables in one predicate should be different, i.e., a hypothesis should not include predicate *P*(*x*,*x*), but may include *P*(*x*,*y*)

5.2. Initial rules and improving search of hypotheses

This section considers another useful sort of background knowledge — a (possibly incorrect) partial initial rule that approximates the concept (rule) to be learned. There are two basic forms of this initial rule: (1) extensional form and (2) intensional form. If a *predicate is defined by other predicates*, we say the definition is **intensional.** Otherwise, a predicate given by example is called **extensional**. It is also possible that background knowledge B contains a predicate in a **mixed** way partially by examples and partially by other predicates. In general, background knowledge presented in a mixed way reduces the search. [44].

Learning using initial extensional rule. An expert or another learning system can provide an initial extensional rule [58]. Then this rule (initial concept) is refined by adding clauses [44]:

1. An algorithm computes the criterion of optimality (usually information gain) of each clause in the *initial concept*.
2. The literal (or conjunction of literals) with the maximum gain is added to the end of the current clause (start clause can be null).
3. If the current clause covers some negative tuples (examples), additional literals are added to rule out the negative tuples.

Learning using initial intensional rules. Next, consider domain knowledge defined in terms of extensional and intensional initial predicates. Systems such as CIGOL [38] make use of (or invent) *background knowledge* of this form. For example, if an extensional definition of the predicate *GrowingStock(x,y,z)* is not given, it could be defined in terms of the intensional predicate GreaterPrice by:

GrowingStock(x,y,z) $\leftarrow$ GreaterPrice(x,y), GreaterPrice(y,z),

where x, y, and z are prices of the stock for days t, t+1, and t+2, respectively.

It is possible that the intensional predicate GrowingStock(x,y,z) added to the hypothesis improves it, but each of predicates GreaterPrice(x,y) and GreaterPrice(y,z) does not improve the hypothesis. Therefore, common search methods may not discover a valuable stock regularity. Pazzani and Kibler [44] suggested that if the literal with the maximum of the optimality criterion (gain) is intensional, then the literal is made extensional and the extensional definition is added to the clause under construction. Note that computation of the optimality criterion, which guides the search, is different for extensional and intensional predicates. For intensional predicates, it is usually involves a Prolog proof. Potentially operationalization can generate very long rules.

Learning using initial intensional and extensional rules. The previous consideration has shown that adding background knowledge can increase the ability of algorithms to find solutions. Table 13 shows an example of partial background knowledge for a stock market forecast. It consists of

- a definition of target predicates Up$_2$(x,y), Up$_3$(x,y,w), Up$_4$(x,y,w,z) with two, three and four arguments to be learned,
- typing information about x,y,w and z,
- intensional predicate Q$_3$(x,y,w) with three arguments to be used for discovering predicate Up$_4$(x,y,w,z), and
- extensional predicates Monday(t) and Tuesday(t) to be used for discovering Up$_4$(x,y,w,z).

Table 13. Partial background knowledge for stock market

Definition of target predicate to be learned:

The learning algorithm should learn the predicates Up$_i$, i.e., generate a logical rule combining i arguments such as Stock(t), Stock(t+1),Stock(t+2) and Stock(t+3)

(1) Up$_2$(Stock(t), Stock(t+1)) $\Leftrightarrow$ (Stock(t)) < Stock(t+1)

(2) Up$_3$(Stock(t), Stock(t+1), Stock(t+2)) $\Leftrightarrow$
(Stock(t)) < Stock(t+1) & (Stock(t+1) < Stock(t+2)

(3) Up$_4$(Stock(t), Stock(t+1), Stock(t+2), Stock(t+3) $\Leftrightarrow$
(Stock(t)) < Stock(t+1) & (Stock(t+1) < Stock(t+2) & Stock(t+2) < Stock(t+3)

Type : UP(float, float, float, float)
Positive examples, Pos: Ex1-- (34.0, 35.1, 36.2, 37.4), Ex2 -- (37, 38.1, 34.4, 35.7)
Negative examples, Neg: Ex3 -- (33.2, 32.1, 33.7, 31.6), Ex4 -- (30.8 29.3, 28.8 27.9)

Intensional Predicate(s):
Q(Stock(t), Stock(t+1), Stock(t+2)) $\Leftrightarrow$ Stock(t+1) - Stock(t) < Stock(t+2) - Stock(t+1)
Type : Q(float, float, float);

Extensional Predicates:
Monday (t). t type: date. This predicate is true for Mondays.
Pos : (04.05.99)(04.12.99)(04.19.99)...(11.01.99)
Tuesday(t). t type: date. This predicate is true for Tuesdays.
Pos : (04.06.99)(04.13.99)(04.20.99)...(11.02.99)

Initial intensional rules for the target concept $Up_4(x,y,w,z)$ are

$Q(Stock(t),Stock(t+1), Stock(t+2) => Up_2(Stock(t), Stock(t+1))$

and

$Q(Stock(t),Stock(t+1), Stock(t+2) => Up_2(Stock(t+1), Stock(t+2))$.

This rule assumes that if growth was accelerated from date t to t+2 then the stock will grow further on date t+3. Background knowledge is called **extended background knowledge** if it includes: (1) extensional knowledge (training examples and extensional predicates), (2) initial rules, and (3) intensional target concept definition.

Pazzani and Kibler [44] found in experiments that *extended background knowledge* with a *correct intensional target* definition avoids exhaustive testing every variable of every predicate and increases the speed of the search. In their experiments, a correct extensional definition of the target concept was found by testing only 2.35% of literals needed for rule discovery if the target concept is not provided. However, the same research has shown that extended background knowledge (1) can *increase the search space*, (2) can *decrease the accuracy* of the resulting hypothesis, if the background knowledge is *partially irrelevant* to the task, and (3) can *increase the number of training examples* required to achieve a given accuracy.

These observations show the need for **balancing initial intensional and extensional predicates** in background knowledge. One of them can be more accurate and can speed up the search for regularity more than other. Therefore, the following procedure will be more efficient:

(1) Compare accuracy of intensional and extensional knowledge.
(2) Include a more accurate one in the background knowledge.
(3) Discover regularities using the most accurate background knowledge from (2).
(4) Discover regularities using all background knowledge.

The modification of this mechanism includes use of **probabilities** assigned to all types of background knowledge. There are several ways to combine extensional and intensional knowledge in discovering regularities. One of them is converting initial rules (predicates) into extensional form (*operationalize a clause*) if it has positive information gain. The extensional predicates are compared to the induced literal with the maximum information gain. This approach is used in the FOIL algorithm. In an **explanation-based learning approach,** the target concept is assumed a **correct**, **intensional definition** of the concept to be learned and the domain knowledge is assumed correct as well. An approach that is more realistic is implemented in algorithms such as FOCL and Discovery. These methods relax the assumption that the target concept and the domain knowledge are correct.

6. Algorithms

A variety of relational machine learning systems have been developed in recent years [33]. Theoretically, these systems have many advantages. In practice though, the complexity of the language must be severely restricted, reducing their applicability. For example, some systems require that the concept definition be expressed in terms of *attribute-value pairs* [9, 32] or only in terms of *unary predicates* [21, 35, 42, 51]. The systems that allow actual *relational concept definitions* (e.g., OCCAM [43], IOE [16], ML-SMART [2]) place *strong restrictions* on the form of induction and the initial knowledge that is provided to the system [44].

Algorithm FOIL [48] learns constant-free Horn clauses, a useful subset of first-order predicate calculus. Later FOIL was extended to use a variety of types of background knowledge to increase the class of problems that can be solved, to decrease the hypothesis space explored, and to increase the accuracy of learned rules.

Algorithm FOCL [44] uses first order logic and FOIL's information-based optimality metric in combination with background knowledge. This is reflected in its full name -- First Order Combined Learner. FOCL has been tested on a variety of problems [45] that includes a domain theory describing when a *student loan* is required to be repaid [46]. It is well known that the general problem of rule generating and testing is NP-complete [22]. Therefore, we face the problem of designing NP-complete algorithms. There are several related questions. What determines the number of rules to be tested? When should one stop generating rules? What is the justification for specifying particular expressions instead of any other expressions? FOCL, FOIL and Discovery use different stop criteria and different mechanisms to generate rules for testing. MMRD selects rules, which are simplest and consistent with measurement scales [31] for a particular task. The algorithm stops generating new rules when the rules become too complex (i.e., statistically insignificant for the data) in spite of the possibly high accuracy of the rules when applied to training data. The obvious other stop criterion is time limitation. FOIL and FOCL are based on the information gain criterion.

Authors of FOCL draw a number of important conclusions about the complexity of learning rules and the value of different sorts of knowledge. Some of these conclusions are summarized below:

- The branching factor grows exponentially in the arity of the available predicates and the predicate to be learned;
- The branching factor grows exponentially in the number of new variables introduced;
- The difficulty in learning a rule is linearly proportional to the number of clauses in the rule;
- *Knowledge about data types provides an exponential decrease for a search* necessary to find a rule;
- Any method (argument constraints, semantic constraints, typing, symmetry, etc.) that eliminates fruitless paths decreases the search cost and increases the accuracy.
- The uniform evaluation function allows FOCL to tolerate domain theories that are

both incorrect and incomplete.

A Discovery system contains several extensions over other RDM algorithms. It permits various forms of **background knowledge** to be exploited. The goal of the Discovery algorithm is to create probabilistic rules in terms of the relations (predicates and literals) defined by a collection of examples and other forms of background knowledge. Discovery as well as FOCL has several advantages over FOIL:

- Limits the search space by using **constraints.**
- Improves the search of hypotheses by using background knowledge with **predicates defined by a rule directly** in addition to predicates defined by a collection of examples.
- Improves the search of hypotheses by accepting as input **a partial, possibly incorrect rule** that is an initial approximation of the predicate to be learned.

There are also advantages of MMRD over FOCL:

- Limits the search space by using the **statistical significance** of hypotheses.
- Limits the search space by using the strength of **data types scales.**
- Shortens the final discovered rule by using the initial set of hypotheses in intensional form directly (without operationalization).

The advantages above represent a way of generalization used in Discovery system. Generalization is the critical issue in applying data-driven forecasting systems. The Discovery method generalizes data through "**lawlike" logical probabilistic rules** presented in first order logic.

Theoretical advantages of Discovery method generalization are based on previous research presented in [53-57, 26-27, 50, 59, see the article in this issue]. The original challenge for Discovery system (it was developed in the frame of the original school in Russia "Machine Methods of Discovery Regularities (MMDR)" [59]) was the simulation of discovering **scientific laws** from empirical data in chemistry and physics. There is a well-know difference between "black box" models and fundamental models (laws) in modern physics. The latter have much longer life, wider scope, and a solid background. There is a reason to believe that Discovery method caught some important features of discovering these regularities ("laws").

7. Experimental results

This section presents some examples of use of Discovery system in the prediction of SP500. Comparison of forecasting performance obtained by use of different methods is presented in three tables below. These data show that Discovery outperformed other methods. Discovery system produced 70% and 84%, respectively, correct up-down and down-up forecasts. The first ARIMA model that was selected without any connections with Discovery system produced a correct buy/sell signal in 62.58% of the cases. The simple Markov process model that was identified by using Discovery system search mechanism gave 79.6%. The third model that exploits parameters prompted by rules discovered by Discovery system gave 75.92%.

The most significant advantage of the first order methods and Discovery system, in particular, is that they can **forecast directly the sign of the difference** in SP500 instead of the value as ARIMA does. ARIMA can generate a sign forecast using a predicted value. The forecast of a value is more complex and available data may not fit for value forecast. The value forecast can be inaccurate and statistically insignificant, but the forecast of the sign can be accurate and statistically significant for the same data. Tables 14 and 15 shows forecast comparison for Discovery system with neural Network and FOIL algorithms.

Table 14. Forecast performance of different methods on test data (% of correct sign (up/down) forecast of SP500C)

Method	1995-1996	1997-1998	Average 1995-1998
Neural network 1 (with preprocessing)	68%	57	62.5%
Rules extracted from NN 1(indirect estimate)	≤68%	≤57%	≤62.5%
Decision tree (Sipina with C4.5 simplification)	67%	60%	64%
First-order logic with probability (Discovery system)	78%	85%	81.5%
First-order logic method (FOIL)	50.50%	45.40%	47.95%

Table 15. Simulated gain per year for SP500 trading (% of initial investment)

Method	*Gain per year in simulated experiments*		
	1995-1996	1997-1998	Average 1995-1998
Adaptive Linear	21.9	18.28	20.09
Discovery system	26.69	43.83	35.26
Buy-and-Hold	30.39	20.56	25.47
Risk-Free	3.05	3.05	3.05
Neural Network	18.94	16.07	17.5

The most interesting is comparison of the Discovery system with the Buy-and-Hold (B&H) strategy. B&H strategy slightly outperformed Discovery system for 1995-1996 (30.39% for B&H and 26.69% for Discovery system. On the other hand, Discovery system significantly outperformed Buy-and-Hold for 1997-1998 (43.83% for Discovery system and 20.56% for B&H.)

8. Conclusion

In this paper we contrasted attribute-value languages (AVLs) with relational languages based on the deterministic first order logic (FOL) and then with stochastic FOL relational languages. This comparison is provided in general and for financial applications and spatial applications. Relational Data Mining is defined as discovering hidden deterministic or stochastic relations in numerical and symbolic data using background knowledge.

The paper provides examples that illustrate: (1) attribute-value representation vs. relational representation, (2) first order logic rules with one and two arguments, (3) the difference between IF-Then first-order logic rules and more traditional IF-Then propositional logic rules, (4) multiattribute representation vs. *joint relational representation*, and

(5) steps of data mining based on joint relational data representation. These examples are derived from financial, image processing domains. The examples intend to clarify a long-term confusion about the difference between logical attribute-value methods and relational methods.

The relational data-mining paradigm is described. It handles uniformly numerical and interval forecasting tasks as well as classification tasks. This paradigm elaborates a single-level data type concept with a rich data structure that consists of elements, relations between elements (predicates) and meaningful operations with elements. Two major ways to express attribute-based examples using predicates are presented based on the predicate invention concept that includes generation of predicates for each attribute or multiple attributes and use of projection functions. We discussed single argument constraints vs. multiple arguments. It is shown that Relational Data Mining (RDM) can handle multiple constrains, initial rules and background knowledge very naturally to reduce the search space in contrast with attribute-based data mining.

Finally, three RMD algorithms (FOIL, FOCL and Discovery) are compared and successful experiential results with financial data using Discovery system are presented. Discovery system contains several extensions over other RDM algorithms. It exploits various forms of background knowledge, estimates of statistical significance of hypotheses, and the relative strength of data type scales.

References

[1] Abu-Mostafa, A., Learning from hints in neural networks. Journal of complexity 6: 192-198, 1990.

[2] Bergadano, F., Giordana, A., & Ponsero, S. (1989). Deduction in top-down inductive learning. Proceedings of the Sixth International Workshop on Machine Learning (pp. 23--25). Ithaca, NY: Morgan Kaufmann.

[3] Blundell, J. Opitz, D.,Object recognition and feature extraction from imagery: the feature analyst® approach, 1st international conference on object-based image analysis (OBIA 2006), Salzburg University, Austria, July 4-5, 2006, http://www.commission4.isprs.org/ obia06/ Papers/09_Automated %20classification%20Generic%20aspects/OBIA2006_Blundell_Opitz.pdf

[4] Bratko, I., Muggleton, S., Varvsek, A. Learning qualitative models of dynamic systems. In Inductive Logic Programming, S. Muggleton, Ed. Academic Press, London, 1992

[5] Bratko, I. Innovative design as learning from examples. In Proceedings of the International Conference on Design to Manufacture in Modern Industries, Bled, Slovenia, June 1993.

[6] Bratko I, Muggleton S (1995): Applications of inductive logic programming. Communications of ACM 38 (11):65-70.

[7] Carnap, R., Logical foundations of probability, Chicago, University of Chicago Press, 1962.

[8] Cramer D. (1998). Fundamental Statistics for Social Research, Step-by-step calculations and computer technique using SPSS for Windows, Routledge, London, NY

[9] Danyluk, A. (1989). Finding new rules for incomplete theories: Explicit biases for induction with contextual information. Proceedings of the Sixth International

Workshop on Machine Learning (pp. 34--36). Ithaca, NY: Morgan Kaufmann.
[10] Dzeroski, S., DeHaspe, L., Ruck, B.M., and Walley, W.J. Classification of river water quality data using machine learning. In: ENVIROSOFT'94, 1994.
[11] Dzeroski S (1996): Inductive Logic Programming and Knowledge Discovery in Databases. In: Advances in Knowledge Discovery and Data Mining, Eds. U. Fayad, G., Piatetsky-Shapiro, P. Smyth, R. Uthurusamy. AAAI, MIT Press, pp. 117-152.
[12] Dzeroski, S., Lavrac, N., Eds., Relational Data Mining, Springer, Berlin, 2001
[13] Džeroski, S., Blockeel, H., Multi-relational data mining 2004: workshop report, ACM SIGKDD Explorations Newsletter, v.6 n.2, December 2004
[14] Fenstad, J.I. Representation of probabilities defined on first order languages // J.N.Crossley, ed., Sets, Models and Recursion Theory: Proceedings of the Summer School in Mathematical Logic and Tenth Logic Colloguium (1967) 156-172.
[15] Flach, P., Giraud-Carrier C., and Lloyd J.W. (1998). Strongly Typed Inductive Concept Learning. In Proceedings of the Eighth International Conference on Inductive Logic Programming (ILP'98), 185-194.
[16] Flann, N., & Dietterich, T. (1989). A study of explanation-based methods for inductive learning. Machine Learning, 4, 187--226.
[17] Fu Li Min (1999) Knowledge Discovery based on Neural Networks, Communications of ACM, v. 42, n 11, pp. 47-50
[18] Getoor , L., Diehl, C, Link mining: a survey, ACM SIGKDD Explorations Newsletter, v.7 n.2, p.3-12, December 2005
[19] Geusebroek, J. M., G.J. Burghouts, J.C. van Gemert, and A.W.M. Smeulders. Invariant representations to prepare for content based image retrieval from first principles. In R. Veltkamp, editor, Trends and Advances in Content-Based Image and Video Retrieval. Springer Verlag, Berlin, 2005. www.science.uva.nl/~jvgemert/pub/Dagstuhl2005Geusebroek.pdf
[20] Halpern J. Y, An analysis of first-order logic of probability. Artificial Intelligence 46: 311-350, 1990.
[21] Hirsh, H. (1989). Combining empirical and analytical learning with version spaces.
[22] Hyafil L, Rivest RL,Constructing optimal binary decision trees is NP-Complete. Information Processing Letters 5 (1),15-17,1976
[23] Kendall M.G., Stuart A. (1977) The advanced theory of statistics, 4th ed., v.1.Charles Griffin & Co LTD, London.
[24] Koller, D., Pfeffer, A., Learning probabilities for noisy first-order rules, In: Proc. of the 15th Int. Joint Conf. on Artificial Intelligence, Nagoya, Japan, 1997
[25] Kovalerchuk B (1973): Classification invariant to coding of objects. Computational systems. 55:90-97, Novosibirsk. (in Russian).
[26] Kovalerchuk B (1975): On cyclical scales. Comp. Syst. 61:51-59, Novosibirsk, Institute of Mathematics (in Russian).
[27] Kovalerchuk, B. (1976), Coordinating methods for decision rules and training data in pattern recognition. Ph. D. Diss., Institute of Mathematics, USSR Academy of Science, Novosibirsk, 146 p. (in Russian).
[28] Kovalerchuk B, Vityaev E, Ruiz JF. (1997). Design of consistent system for radiologists to support breast cancer diagnosis. Joint Conf. of Information Sciences,

Duke University, NC, 2: 118-121, 1997.

[29] Kovalerchuk B, Vityaev E (1998): Discovering Lawlike Regularities in Financial Time Series. Journal of Computational Intelligence in Finance 6 (3):12-26.

[30] Kovalerchuk B., Vityaev E. Data Mining in Finance: Advances in Relational and Hybrid methods. (Kluwer international series in engineering and computer science; SECS 547), Kluwer Academic Publishers, 2000, p.308

[31] Krantz DH, Luce RD, Suppes P, and Tversky A: Foundations of Measurement V.1-3, Acad. Press, NY, 1971, 1989, 1990.

[32] Lebowitz, M. (1986). Integrated learning: Controlling explanation. Cognitive Science, 10.

[33] Mitchell (1997): Machine Learning, Prentice Hall.

[34] Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based learning: A unifying view. Machine Learning, 1, 47--80.

[35] Mooney, R., & Ourston, D. (1989). Induction over the unexplained: Integrated learning of concepts with both explainable and conventional aspects. Proc. 6th International Workshop on Machine Learning (pp. 5--7). Ithaca, NY: Morgan Kaufmann.

[36] Muggleton S. (1994) Bayesian inductive logic programming. In Proceedings of the Eleventh International Conference on Machine Learning W. Cohen and H. Hirsh, Eds., pp. 371–379.

[37] Muggleton S (1999): Scientific Knowledge Discovery Using Inductive Logic Programming, Communications of ACM, vol. 42, N11, pp. 43-46.

[38] Muggleton, S., & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. Proceedings of the Fifth International Workshop on Machine Learning (pp. 339--352). Ann Arbor, MI: Morgan Kaufmann.

[39] Muggleton, S., King, R.D. and Sternberg, M.J.E. (1992) Protein secondary structure prediction using logic. Prot. Eng. 5, 7), 647–657

[40] Mooney, R. J. , P. Melville, L. P. Rupert Tang, J. Shavlik, I. de Castro Dutra, D. Page, and V. Santos Costa. Relational data mining with inductive logic programming for link discovery. In Proceedings of the National Science Foundation Workshop on Next Generation Data Mining, Baltimore, Maryland, USA, 2002.

[41] Luc De Raedt, From Inductive Logic Programming to Multi-Relational Data Mining, Springer.2006

[42] Pazzani, M. (1989). Explanation-based learning with weak domain theories. Proceedings of the Sixth International Workshop on Machine Learning (pp. 72-- 74). Ithaca, NY: Morgan Kaufmann.

[43] Pazzani, M. J. (1990). Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods. Hillsdale, NJ: Lawrence Erlbaum Associates.

[44] Pazzani, M., Kibler, D. (1992). The utility of prior knowledge in inductive learning. Machine Learning, 9, 54-97

[45] Pazzani, M., (1997), Comprehensible Knowledge Discovery: Gaining Insight from Data. First Federal Data Mining Conference and Exposition, pp. 73-82. Washington, DC

[46] Pazzani, M., Brunk, C. (1990), Detecting and correcting errors in rule-based expert systems: An integration of empirical and explanation-based learning. Pro-

ceedings of the Workshop on Knowledge Acquisition for Knowledge-Based System. Banff, Canada.

[47] Pfanzagl J. (1971). Theory of measurement (in cooperation with V.Baumann, H.Huber) ed. Physica-Verlag.

[59] Plaza, A. Martinez, P. Plaza, J. Perez, R. Spatial/Spectral analysis of hyperspectral image data, In: 2003 IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data, 2003, pp. 298 – 307.

[48] Quinlan, J. R. (1990). Learning logical definitions from relations. Machine Learning, 5, 239-266.

[49] Russel S, Norvig P (1995): Artificial Intelligence. A Modern Approach, Prentice Hall.

[50] Samokhvalov, K., (1973). On theory of empirical prediction, Comp. Syst., #55, 3-35. IM, Novosibisk (in Russian)

[51] Shavlik, J., & Towell, G. (1989). Combining explanation-based learning and artificial neural networks. Proceedings of the Sixth International Workshop on Machine Learning , pp. 90-93. Ithaca, NY: Morgan Kaufmann.

[52] Seventh International Workshop on Multimedia Data Mining, "Merging Multimedia and Data Mining Research", MDM/KDD2006 , 2006, Philadelphia, USA, http://www.fortune. binghamton.edu/MDM2006/ Multimedia %20Data%20Mining%20_%20KDD%2705 _files/content_data/MDMKDD2006-v2.pdf

[53] Vityaev E. (1976). Method for forecasting and discovering regularities, (Computing systems, #67), Institute of Mathematics, Novosibirsk, pp.54-68 (in Russian)..

[54] Vityaev E. (1983). Data Analysis in the languages of empirical systems. Ph.D. Diss, Institute of Mathematics SD RAS, Novosibirsk, p.192. (In Russian)

[55] Vityaev E. (1992) Semantic approach to knowledge base development: Semantic probabilistic inference. (Comp. Syst. #146): 19-49, Novosibirsk. (in Russian).

[56] Vityaev E., Moskvitin A. (1993). Introduction to discovery theory: Discovery software system. (Comp. Syst., #148): 117-163, Novosibirsk. (in Russian).

[57] Vityaev E., Logvinenko A. (1995). Method for testing systems of axiom, Computational Systems, Theory of computation and languages of specification, (Comp. Syst., #152), Novosibirsk, p.119-139. (in Russian).

[58] Widmer, G. (1990). Incremental knowledge-intensive learning: A case study based on an extension to Bergadano & Giordana's integrated learning strategy (Technical Report). Austrian Research Institute for Artificial Intelligence.

[59] Zagoruiko N.G., Elkina V.N. Eds. (1976), Machine Methods for Discovering Regularities. Proceedings of MOZ'76, Novosibirsk. (In Russian)