In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

# Ryszard S. Michalski: The Vision and Evolution of Machine Learning

Janusz Wojtusiak[1], Kenneth A. Kaufman[1]

[1] Machine Learning and Inference Laboratory
George Mason University
Fairfax, VA 22030, USA
jwojt@mli.gmu.edu, ken.kaufman@gmail.com

**Abstract.** Ryszard S. Michalski was an outstanding scientist who dedicated his life to research and discovery. He pioneered so many areas and methods of machine learning that it is not possible to describe them properly in one chapter. Thus, we present a brief summary of what we believe are the most important aspects of his research, and present the vision of machine learning that he communicated to us on multiple occasions. The most important topics mentioned in this chapter are: natural induction, knowledge mining, AQ learning, conceptual clustering, VL1 and attributional calculus, constructive induction, the learnable evolution model, inductive databases, methods of plausible reasoning, and the inferential theory of learning.

**Keywords:** Machine Learning, Natural Induction, Concept Learning, Conceptual Clustering, Knowledge Mining

**This chapter is dedicated to Dr. Ryszard S. Michalski, our mentor and friend, who opened for us the wonderful world of machine learning.**

## 1 Introduction

The very fast growth of machine learning and related disciplines is attributed to its great importance and applicability in almost all aspects of modern society. Learning is considered one of the most important aspects of intelligent systems, thus work in this area has been attempted since the very beginning of the field of artificial intelligence over half a century ago. Dr. Ryszard S. Michalski was among the most significant contributors to machine learning. His involvement in the formation of the field, including organization of the first workshops and conferences on machine learning, the first books in the area, including [29][30], is only a small part of the overall contributions.

In this chapter we overview selected research areas originated by Dr. Michalski, and present the vision and directions of the field presented to us over many years of work with him. We understand that it is not possible to summarize four decades of

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

intense research and over 350 publications in one chapter; thus, we emphasize the newest aspects of his research, and those we believe are the most significant.

We start this chapter with an overview of natural induction, an approach to inductive learning that puts a human recipient of learned knowledge in a central position. A key part of natural induction concerning representation of knowledge is described in Section 3. We also discuss learning methods and paradigms originated by Dr. Michalski, including the learnable evolution model, and inductive databases. We conclude this chapter by describing his selected contributions to theory of machine learning, and describe software built to teach and entertain.

## 2 Natural Induction

Before describing selected methods and algorithmic contributions of Ryszard Michalski to the field of machine learning, let us start by describing *natural induction*, which can be viewed as a central aspect of his research over the past decades. The actual name "natural induction" did not appear until the 1990s, but its principles can be found in different forms back at least to the early 1970s. The name natural induction consisting of two parts suggests that induction is done in a natural way. The induction, or more precisely inductive reasoning or learning, is a falsity preserving reasoning process in which generated knowledge is generated from observations/examples and background knowledge. The word *natural* relates to the knowledge resulting from learning that is in a form easy to understand by people. Thus, natural induction refers to an inductive learning process whose results are designed to be natural to people.

There are several forms of knowledge that are natural to people, including natural language descriptions [43], easy to interpret rules, graphical representations, relatively small decision trees, Bayesian networks, etc. These forms of knowledge are considered to be "transparent box", in contrast to "black box" representations that may provide very good predictions, but be hard to understand. The latter can be exemplified by neural networks, random forests, support vector machines, and many other models often used in machine learning.

In order to present results of reasoning in forms easily understandable, one can either transform the learned knowledge, or learn directly in a language natural to people. While the former can be exemplified by all kinds of visualization methods, the latter requires reasoning directly in some language that corresponds directly to natural language statements. This and other issues regarding representation of knowledge are described in next section.

One important representation of knowledge that directly corresponds to natural language is an attributional rule, described in section 3, and exemplified in Figure 1. The rule can be paraphrased: "*The plan to do is to run experiments, if it is a weekend, the weather is rainy and cold, the available workstation has a clock speed at least 2GHz, and the available laboratory is lab1 or lab2, except for when there is a server malfunction.*" The value "weekend" is a higher-level value of a structured (hierarchical) attribute "Day", the attribute "Weather" is a *compound attribut*e that takes a conjunction of values. The pairs of numbers within conditions represent

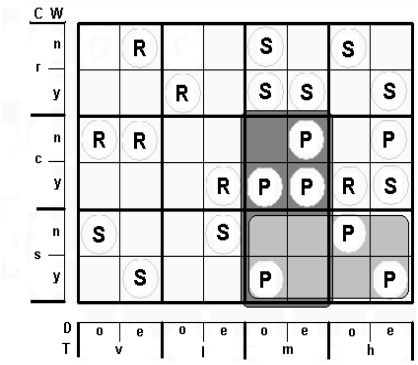
In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

numbers of positive and negative examples satisfying these conditions. The entire rule is satisfied by 27 positive and 1 negative example, as indicated by "p=27, n=1" in the rule's annotation, and its quality, defined by (6), is 0.9.

```
[Plan to do = run experiments]
    [Day = weekend: 38,131] &
    [Weather = rainy and cold: 22, 5] &
    [Available workstation clock speed >= 2GHz: 50,5] &
    [Available lab = lab1 or lab3: 43,57]
  └ [Server malfunction: 0,3] : p=27, n=1, Q=0.9
```

**Fig. 1.** An example of an attributional rule.

Two graphical representations of knowledge developed by Dr. Michalski are *general logic diagrams* (GLD), and *concept association graphs* (CAG). In GLDs, each cell represents a combination of input attributes, and examples are marked within cells by their class [19]. Complexes (see section 3) are represented by rectangles. An example of a GLD with 19 examples and 2 complexes is presented in Figure 2. A CAG represents rules as a labeled graph with varying thicknesses of the links indicating the strength of the knowledge component represented by the link [11]. An example is presented in Figure 3.




C – Condition, W – Wind, T – Temperature, D – Daytype, r – rain, c – cloudy, s – sunny, n – no, y – yes, v – very low, l – low, m – medium, h – high, o – workday, e – weekend, P – Play, R – Read, S - Shop

**Fig. 2.** A general logic diagram (reproduced from [34])**.**

Natural induction is an important part of *knowledge mining*, an approach to data analysis that is used to derive high level concepts from data and background knowledge [13]. In contrast to traditional data mining, this approach is not limited to analyzing large amounts of data. To the contrary, knowledge mining can analyze

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

both large and small amounts of data and, more importantly, use problem-oriented background knowledge. It can be viewed as a next step following in the evolution of data analysis and knowledge creation methods.

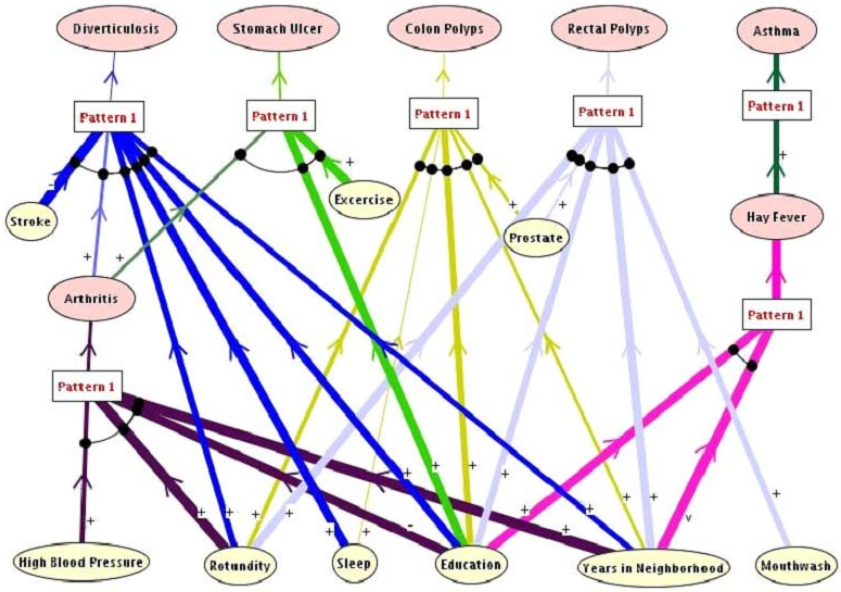



**Fig. 3.** A concept association graph representing seven patterns in a medical database (reproduced from [34]).

Natural induction requires human-oriented representation of knowledge and learning algorithms that can utilize power of that representation. In the next two sections we describe the creation of attributional calculus, a representation language ofr natural induction, and learning methods that can utilize this language.

## 3 Representation

Appropriate representation of knowledge and data is critical for successful application of machine learning methods. To go along with his nascent $A^q$ algorithm, Michalski [20] developed the VL1 knowledge representation language, a variable-valued logic with an expressive power somewhere between those of propositional and predicate calculus. The availability of such internal operators as range and internal disjunction made it suitable for representing compact, easily understood rulesets. Contrast that with, for example, the traditional-style decision trees that still persist to

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

this day, in which several levels of nodes may be needed to subdivide a single continuous attribute into the ranges needed for the learned classifier.

But to take advantage of the power of such a representation, it was necessary to have available all data types that mirrored our natural cognitive experience – nominal, discrete-ordered, hierarchically structured, cyclic, continuous, set-valued, etc. – and be able to operate upon and reason with them as needed. Thus, as part of the AQ packages developed and distributed over the years (see section 4.1), the operators to handle the calculus of (most of) these data types at all stages – input, extension against, selection, trimming, and output – was built in.

This would be enough for many applications, but many was not enough. The representation of knowledge soon included annotations, so that it would be easy to access and work with and from the metaknowledge as well as the basic discovered facts. Now it would be easy to access the support and confidence levels of different knowledge components, and use these to guide your data exploration process.

The VL1 language was subsequently extended to the VL2 representation language [16] that added the existential quantifier. Now the natural induction schema extended to structured descriptions, in which components of the whole and relations among them could be understandably expressed; such descriptions would be utilized in the INDUCE programs for learning structured descriptions (see section 4.2), the CLUSTER conceptual clustering programs (see section 4.5), and the SPARC program for qualitative sequence analysis, discovery and prediction (see section 4.3).

In 2004, Michalski presented a further extension of these languages, *Attributional Calculus* [26]. Attributional calculus further simplified representation of concepts we use naturally by introducing constructs for counting attributes (e.g., at least 3 of these 4 conditions must be present) and relationships among attributes (e.g., height < length + width). Another important idea formalized in attributional calculus concerns using unknown, not-applicable, and irrelevant meta-values. Although all three values appeared in different forms in the early work of Dr. Michalski, it is in attributional calculus where their semantics are first described together. Unknown meta-values, are the most popular, and represent situations in which a regular value exists, but it is not recorded for some reason. Not-applicable meta-values are used when the regular values do not exist. For example the value of an attribute describing pregnancy in male patients does not exist, thus it is not applicable. The irrelevant meta-value is used when a regular value exists, but it is considered not to be relevant to the current learning task. The two latter types of meta-values constitute problem background knowledge.

In attributional calculus, learned knowledge is represented in the form of *attributional rules,* which consist of *attributional conditions*. An attributional condition takes the form (1):

$$[L \; rel \; R{:}\; A] \tag{1}$$

where $L$ is an attribute, an internal conjunction or disjunction of attributes, a compound attribute, or an expression; *rel* is one of $=, >, <, \leq, \geq, :,$ or $\neq$; $R$ is an attribute value, an internal disjunction of attribute values, an attribute, an internal conjunction of values of attributes that are constituents of a compound attribute, or an expression, and A is an optional annotation that may list |p| and |n| values for the

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

condition, defined as the numbers of positive and negative examples, respectively, that satisfy the condition, the condition's consistency defined as |p|/(|p|+|n|), and other measures of conditions' quality. There are several different forms of attributional rules allowed by attributional calculus. Three important forms of attributional rules are (2) - (4).

$$\text{CONSEQUENT} \Leftarrow \text{PREMISE} \tag{2}$$

$$\text{CONSEQUENT} \Leftarrow \text{PREMISE} \sqsubset \text{EXCEPTION} \tag{3}$$

$$\text{CONSEQUENT} \Leftarrow \text{PREMISE} \sqsupset \text{PRECONDITION} \tag{4}$$

where PREMISE, CONSEQUENT, EXCEPTION, and PRECONDITION are *complexes*, that is, conjunctions of attributional conditions. An EXCEPTION can also be an explicit list of examples that constitute exceptions to the rule. The rules (2) are interpreted that the CONSEQUENT is true whenever the PREMISE is true. The rules (3) are interpreted that the CONSEQUENT is true whenever the PREMISE is true, except for when the EXCEPTION is true. The rules (4) are interpreted that the CONSEQUENT is true whenever the PREMISE is true, provided that the PRECONDITION is true. Each rule may be optionally annotated with several parameters such as numbers of covered examples (positive and negative), the rule complexity etc.

Static representation used by *selective induction* methods (most inductive learning methods including rule learners, decision tree learners, Bayesian networks, etc.) did not seem sufficient for many applications. This sparked the development of *constructive induction*, in which a representation space is changed to better fit the machine learning problem being considered [22]. This change of representation space may include removing irrelevant attributes, adjusting discretization of existing attributes, including operations on hierarchies, and most importantly the construction of new attributes. The latter can be in the form of arithmetic or logic expressions, counting attributes (generalization of quantifiers), or some other special forms. A general classification of constructive induction methods include data-driven constrictive induction (DCI) in which modifications of the representation space are based on the analysis of input data, hypothesis-driven constructive induction (HCI) in which modifications of the representation space are based on the analysis of learned hypotheses, knowledge-driven constructive induction (KCI) in which modifications to the representation space are based on background knowledge provided to the system, and multistrategy constructive induction (MCI) that dynamically selects appropriate DCI, HCI, and KCI methods based on a learning problem's characteristics. Some recent work on constructive induction includes the investigation of different specificities of advices provided to the system and the use of constructive induction in intelligent evolutionary optimization [50].

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

# 4 Learning

This section gives a brief overview of machine learning methods developed or co-developed by Ryszard Michalski over the past four decades. These methods were implemented in numerous computer programs. Most of them follow a general separate and conquer approach to inductive learning. They take data and background knowledge and generate new knowledge, often in the form of attributional rules.

## 4.1 AQ: Learning Attributional Rules from Examples

The well-known family of AQ programs originated with the $A^q$ algorithm for solving the general covering problem [14][18]. In fact, the AQ programs perform a simplified version of the original algorithm. AQ programs pioneered the separate and conquer approach to rule learning [6]. Numerous implementations and extensions of the method were developed using different programming languages and on different platforms over the years. Among the best known AQ implementations are AQ7 [26], AQ11 [38], AQ15c [47], AQ17 [1], AQ19 [32], and most recently AQ21[53].

The general problem to which AQ programs are applied is learning from examples. Given a set of examples $e_1$, $e_2$, .. $e_n$, belonging to classes $c_1$, …, $c_k$, find a set of rules (5), where $C_{i,j}$ are complexes, that describe the input data. Note that this is the form (2) of rules. Some AQ implementations are also able to create rules in the forms (3) and (4) that include exceptions and preconditions.

$$\begin{array}{c} c_1 \Leftarrow C_{1,1},\ c_1 \Leftarrow C_{1,2},\ \ldots,\ c_1 \Leftarrow C_{1,t1} \\ c_2 \Leftarrow C_{2,1},\ c_2 \Leftarrow C_{2,2},\ \ldots,\ c_2 \Leftarrow C_{2,t2} \\ \ldots \\ c_k \Leftarrow C_{k,1},\ c_k \Leftarrow C_{k,2},\ \ldots,\ c_k \Leftarrow C_{k,tk} \end{array} \quad (5)$$

A basic version of the AQ learning algorithm is presented in Figure 4. It takes as input a set of positive examples of a concept, P, and set of negative examples, N, belonging to all other classes (examples not belonging to the learned concept), and a quality measure. It returns a complete and consistent hypothesis in the form of an *attributional ruleset*, optimized according to the given *lexicographical evaluation functional* (LEF). In the case when input data consists of multiple classes, learning is repeated for each class against all other classes.

```
Hypothesis = null
While P is not empty
 Select a seed example e from P
 Generate star G(e, N)
 Select the best rule R from G(e, N) according to LEF, and
   include it in Hypothesis
 Remove from P all examples covered by the selected rule
Return learned Hypothesis
```

**Fig. 4.** Basic AQ algorithm.

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

The central part of the algorithm is the generation of star *G(e, N)* given a seed *p* and set of negative examples *N*. The star is a set of maximally general rules covering the seed *e*, but not covering any negative example from *N*. A star is constructed by intersecting partial stars which are generated using the *extension-against* operator [22]. The extension-against that takes two data points and creates a set of single-condition rules. The rules are maximal generalizations of one data point (the seed) that does not cover the second data point (a selected negative example). The set of rules created by extension-against is called a *local star*. An intersection of local stars for all negative examples is the star *G(e, N)*. To narrow down a possibly very large number of intermediate generalizations, AQ uses a beam search that at each step of star generation keeps no more than a parameter-defined number of best rules, as determined by the given pattern quality measure, as defined by the LEF. LEF evaluates rules through a sequence of criteria with defined thresholds. Different AQ programs have implemented different LEF criteria, but most of them select by default rules that cover the most positive examples and are the simplest. Many other criteria have been introduced that can be tuned to a specific problem by the user. A complete list of LEF criteria available in the AQ21 system is presented in [48].

To select a rule from a star, the algorithm also uses a LEF, but possibly with different criteria than for the beam search. One important extension of the method is to learn approximate theories from noisy data, and discover strong regularities, patterns, in the data. To do so, AQ at each step of star generation is able to optimize rules according to the $Q(w)$ rule quality measure. The measure is given by (6) and has been defined by Michalski and Kaufman [33].

$$Q(w) = cov^{w} * consig^{1-w} \tag{6}$$

where

$$cov=|p| / |P| \tag{7}$$

and

$$consig=((|p| / (|p| + |n|)) - (|P| /(|P| + |N|))) * (|P| + |N|) / |N| \tag{8}$$

Here, $|p|$ and $|n|$ are the numbers of positive and negative examples covered by the rule, and $|P|$ and $|N|$ are the numbers of positive and negative examples in the training dataset, respectively. The $Q(w)$ definition (6) assures in practice that $consig \geq 0$ which means that a rule's prediction is better than a blind guess of the target class. The actual implemented formula handles also the case when $consig < 0$, but such rules are always dropped as not useful.

Optimization of rules consists of several possible operations which may lead to improvement of rules' quality. The operations are: abstraction of conditions, specialization of conditions, removal of conditions, and removal of entire rules.

Numerous other modifications and extensions of the AQ learning algorithm have been made over years. For example, some AQ versions use several seeds (to protect the method against noise), employ different concept representations (attributional or relational), generate rules with different interrelationships (independent, disjoint or

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

sequentially ordered covers), use different methods for handling data inconsistency (minimum, maximum, free and statistic-based generalization), learn rules in a batch of incremental mode, seek rules that represent the best trade-off between their consistency, coverage and simplicity, use different criteria of rule optimality, involve operators for deriving more relevant attributes (data-driven, hypothesis-driven or multistrategy constructive induction), apply prior knowledge (a- and l-rules, knowledge-driven constructive induction), post-optimize learned descriptions (TRUNC/s, TRUNC/sg and TRUNC/nl), generate single or alternative descriptions, learn rules with exceptions or preconditions, handle unknown, not-applicable and irrelevant, meta-values, learn in standard multiclass mode or without a contrast set, etc.

The evolution of AQ-based learning systems was not linear, and no one system had encompassed all of these methodological features. In recent years, Dr. Michalski along with his group of collaborators initiated an attempt to implement most of the AQ learning methodology developed over many years into one system, AQ21. Its goal is to become a laboratory for natural induction, that is, to allow users to learn attributional rules of different types, present results in easy to interpret forms, such as natural language, and allow experimentation exploiting large numbers of parameters to specify users' preferences. The program, which is being constantly extended, is the most advanced AQ implementation to date, and this effort continues at the GMU Machine Learning and Inference Laboratory.

### 4.2 INDUCE: Learning Structures

Learning structural descriptions in the form of easy to understand rules is realized by a class of programs called INDUCE. In contrast to AQ programs, INDUCE does not take input in the form of labeled examples, but rather in the form of sets of rules describing considered objects. Its goal is to arrive at a smaller set of rules consistent with original ones. The INDUCE programs are multipurpose and applicable to a wide class of problems, but they are particularly suitable for learning structural descriptions of classes of objects in which the number of relations between objects is not known in advance. The method takes as an input a set of rules (9) describing objects in classes $c_1, c_2, \ldots, c_n$.

$$\begin{array}{c} c_1 \Leftarrow C_{1,1},\ c_1 \Leftarrow C_{1,2},\ \ldots,\ c_1 \Leftarrow C_{1,r1} \\ c_2 \Leftarrow C_{2,1},\ c_2 \Leftarrow C_{2,2},\ \ldots,\ c_2 \Leftarrow C_{2,r2} \\ \ldots \\ c_k \Leftarrow C_{k,1},\ c_k \Leftarrow C_{k,2},\ \ldots,\ c_k \Leftarrow C_{k,rk} \end{array} \qquad (9)$$

The goal is to fined a set of rules (10) where $ri \geq ti$, $i=1..k$. The obtained set of rules is not only more compact (fewer rules), but it is also a generalization of input rules.

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

$$\begin{array}{c} c_1 \Leftarrow C_{1,1},\ c_1 \Leftarrow C_{1,2},\ \ldots,\ c_1 \Leftarrow C_{1,t1} \\ c_2 \Leftarrow C_{2,1},\ c_2 \Leftarrow C_{2,2},\ \ldots,\ c_2 \Leftarrow C_{2,t2} \\ \ldots \\ c_k \Leftarrow C_{k,1},\ c_k \Leftarrow C_{k,2},\ \ldots,\ c_k \Leftarrow C_{k,tk} \end{array} \quad (10)$$

The INDUCE method follows the star generation algorithm similar to one used in AQ, but in addition to extension-against, it uses other generalization operators [22]. It is more general than the AQ example learner, but is also less efficient. Because most real word learning problems can be represented in the in the form of labeled examples, development of AQ systems has progressed much further than that of INDUCE.

### 4.3 SPARC: Learning Sequences

Another type of learning is done through *part-to-whole generalization* in which given observations of a part of an observed set of objects, the goal is to hypothesize the entire set of objects. One specific case of this type of learning concerns learning and predicting sequences (ordered sets) of objects. Among the many real world applications of predicting sequences are predicting stock market behavior, predicting outcomes of medical treatments, predicting gene sequences, predicting computer users' behavior, and so forth.

The problem considered here is to predict events (objects) that follow a given sequence of objects. Given the sequence of events in the form (11), the goal is to find events $e_{n+1}$, $e_{n+2}$, and so on.

$$e_1, e_2, e_3, e_n \quad (11)$$

Each of the objects $e_i$ may be described in terms of one or more attributes (12), where $v_{i1}$, $v_{i2}$, .. $v_{ik}$ are values of attributes $X_1$, …, $X_k$ in the event $e_i$. These attributes may be of different types, e.g. nominal, structured, ordinal, cyclic, ratio.

$$e_i = (v_{i1}, v_{i2}, \ldots, v_{ik}) \quad (12)$$

In some specific situations it may not be necessary to predict entire events $e_{n+1}$, $e_{n+2}$, …, but rather values of selected attributes in these events. It may also be appropriate to predict not one, but several plausible events that fit into the sequence in a particular place. In this *non-deterministic prediction problem*, the goal is to discover some properties of events in the sequence, not necessarily precisely predict them [3].

Computer programs called SPARC represent a multistrategy approach to discovering sequences, by combining the *periodic conjunctive model*, *lookback decomposition model*, and *disjunctive normal form* models. Two programs were created, SPARC/E, a specialized system designed for playing the eleusis card game [3][35], and the SPARC/G general purpose sequence discovery tool [36]. The programs work in three steps. In the first step, constructive induction methods are

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

applied to derive attributes that may better characterize sequences of events. In the second step, rules characterizing sequences are derived by applying AQ learning to data prepared for each type of model. In the last step, final rules are selected based on their consistency and simplicity. These rules can be later used to predict plausible continuationa of the input sequence.

### 4.4 ABACUS: Learning Equations

Quantitative discovery deals with discovering equations from data. The ABACUS approach to quantitative discovery is able to incorporate symbolic attributes in learning equations, cope with irrelevant attributes, and work with noisy data. Its main advantage is that it is able do discover multiple equations, each characterizing a subset of data, and provide preconditions under which the equations should be used [5].

The ABACUS method works in two steps. First, in the *equation discovery module*, it analyzes input data and creates equations. If more than one equation is needed, the data is split into subsets. Because of the enormous size of the possible search space of possible equations ABACUS uses a number of heuristics. One such heuristic involves constructing and searching proportionality graphs of attributes that are *qualitatively proportional* (or *inversely quantitatively proportional*). For such attributes, a given percentage of values of one attribute grows (or decreases) with values of other attribute for other attributes constant. Other heuristics include checking for compatibility of units, detection of redundancy in formulas, and tautology detection. In the second step, a *precondition generation module* generates logic preconditions characterizing the subsets of data for which equations were created in the first step. To do so, ABACUS employs the AQ learning method.

The system was applied to multiple datasets, and was able to correctly discover, for example, Stoke's law and the law of conservation of kinetic energy.

### 4.5 CLUSTER: Conceptual Clustering

Learning from observation is, after concept learning from examples, the second largest class of problems in machine learning. The goal is to construct meaningful classes from observed objects. Traditional clustering methods group objects based on their similarity, without regard to what global concepts they form. The answer to this problem was proposed by Michalski and is called *conceptual clustering* [21][41] in which objects are grouped together based on their *conceptual cohesiveness* (13). This measure is based not only on the distance between objects, but more importantly on the set of concepts, $C$, that are available for describing the objects, and other objects, $E$, in the dataset.

$$\textit{conceptual cohesiveness}(O_1, O_2) = \mathrm{f}(O_1, O_2, C, E) \tag{13}$$

An example of a problem in which standard distance-based clustering methods may incorrectly form clusters is presented in Figure 5. The two marked points are the

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

closest objects to one another in the figure, so they would quickly and invariably be assigned to he same cluster by many methods. In contrast, a conceptual clustering method that has been provided with the concepts of letters and their shapes would be able to correctly create two clusters representing the X and the Y.

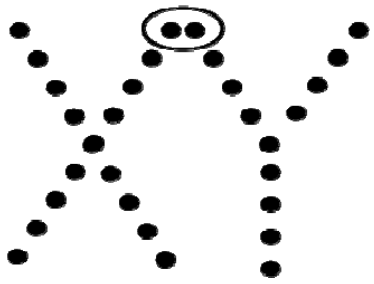


**Fig. 5.** An example of a conceptual clustering problem.

The basic algorithm used in CLUSTER implementations of conceptual clustering takes as an input a set of objects, *E*, the desired number of clusters, *k*, and an LEF expressing criteria of clusters' quality. It starts by randomly selecting *k* seed examples, $E_s=\{e_1, \ldots, e_k\}$. For each seed example $o_i$ a star $G(o_i, E_s\backslash\{o_i\})$ is generated. Then a clustering is created by selecting disjoint rules from the stars according to the LEF. If termination criteria are not met, a new set of seeds is selected from examples covered by rules in the clustering, and the procedure is repeated [40].

Recent research on conceptual clustering concerns building goal-oriented clusters from the perspective of different *viewpoints* [45]. A viewpoint is defined as a subset of attributes in the representation space that can be meaningfully combined. For example, one may cluster students in a database from the viewpoint of their intellectual abilities, while other may want to cluster students based on their demographics. These two viewpoints require clearly different sets of attributes, which are drawn from the set of all attributes available in database.

## 5 Learnable Evolution Model

The learnable evolution model (LEM) is an evolutionary optimization method that employs machine learning to direct the evolutionary process [24][25]. Specifically, LEM creates general hypotheses indicating regions in the search space that likely contain optimal solutions and then instantiates these hypotheses to generate new candidate solutions. In order to apply concept learning, LEM creates two groups of individuals that are respectively high- and low-performing according to the fitness function being optimized. These individuals can be selected from the current population or a combination of current and past populations of individuals. The group of high-performing individuals is called H-Group and the group of low-

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

performing individuals is called L-Group. Once the groups are selected, LEM applies concept learning to create a general hypothesis describing the H-Group in contrast to the L-Group. The hypotheses are then instantiated to create new candidate solutions. In the final step, a new population is assembled from old and new individuals, and the process is repeated until stopping criteria are met.

Very successful initial implementations of the learnable evolution model sparked development of the third generation of LEM software, called LEM3. It extends many ideas found in the original LEM methodologies, some of which are unique in the field of evolutionary computation. The general flow diagram of LEM3's algorithm is presented in Figure 6. In addition to components found in standard evolutionary computation methods, such as generation of an initial population, evaluation of individuals, and selection of individuals, LEM3 includes several novel components. It dynamically selects one or more *innovation* methods to create new individuals. These methods are: *learn & instantiate*, the aforementioned main mechanism for creating new individuals in LEM3; *probe*, to apply traditional operators such as mutation and crossover, *search locally*, to apply a user-defined local search method; and *randomize*, to add to the current population a number of randomly created individuals, or restart the evolutionary process. One of the major novelties of LEM3 is the ability to automatically adjust representation space through constructive induction [49][50].

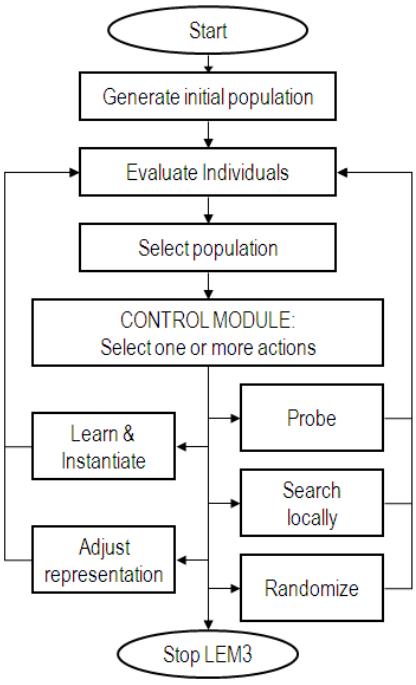



**Fig. 6.** Diagram of the LEM3 algorithm (reproduced from [49]).

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

Theoretical and experimental work indicates that LEM is particularly suitable for optimization problems in which the fitness evaluation is costly. This is because of the trade-off between significantly shorter evolution length [25][52], and more complex learning and instantiation when compared to simple operators used in evolutionary computation. Moreover, the use of machine learning to guide evolutionary computation extends the applicability of LEM. For example, because of the use of AQ21 as a learning module in LEM3, it is able to handle optimization problems naturally described using different types of attributes (nominal, structured, ordinal, cyclic, interval, ratio, and compound) and background knowledge provided to the learning program [51].

A class of LEM-based systems for heat exchanger optimization has been developed. These include the ISHED system for optimizing evaporators and ISCOD system for optimizing condensers [4][10]. These specialized systems combine LEM's learning and instantiation operators with specialized probing operators that are specifically designed to work with heat exchangers. Based on the ISHED and ISCOD systems, Michalski and Kaufman proposed a general LEMd methodology for optimizing complex systems [31].

## 6 Inductive Databases and Knowledge Scouts

Imagine a database system that can answer queries about items not present in the database. By applying inductive learning to data and background knowledge stored in the database an *inductive database* is able to answer such queries. In fact, an inductive database system should integrate many inductive and deductive reasoning methods on data and knowledge stored in the database. Results of reasoning are added to the existing knowledge in the database and can be reused when answering further queries. The latter feature distinguishes the concept of *inductive databases* proposed by Michalski and his collaborators, from those often found in literature. The key idea behind inductive database is one of *knowledge system* that combines database and knowledge base. The term knowledge system signifies integration of database and a relevant knowledge base to support knowledge mining and knowledge application. Knowledge bases contain knowledge created by experts and knowledge generated by inductive learning operators applied to data in the database and to knowledge in the knowledge base.

The idea of inductive database originated in 1980s through development of computer systems QUIN [28][46], ADVISE [27], and AURORA [7]. It was later extended in the inductive database system INLEN (INference and LEarNing) [7][14], and most recently in its successor, VINLEN [12][15]. Both INLEN and VINLEN provide interfaces that allow users to access knowledge system and apply knowledge generation and application operators, as depicted in Figures 7 and 8.

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

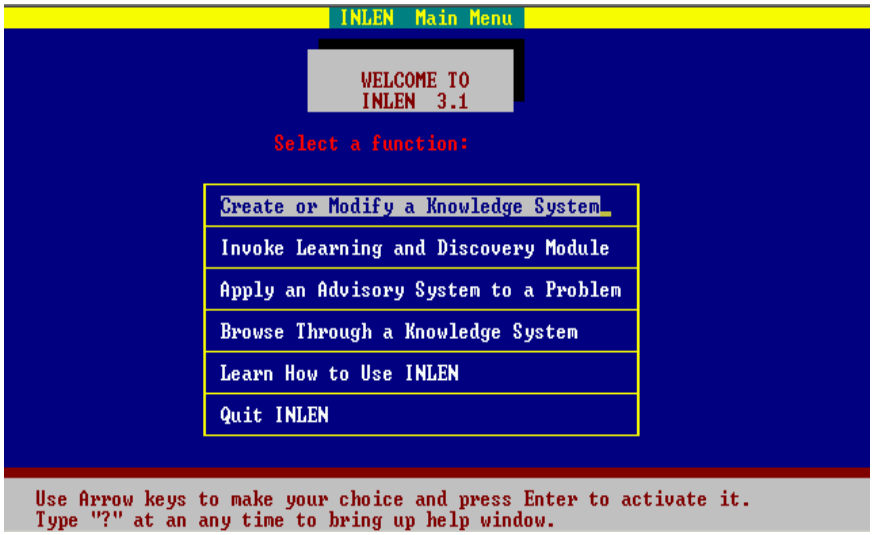



**Fig. 7.** The INLEN system main menu.

A knowledge scout is a script that allows intelligent search for answers in an inductive database [11]. It usually involves multiple steps of inductive reasoning and knowledge application in which one step depends on results of previous steps. One script language that can create knowledge scouts is *knowledge generation language* (KGL) developed within the INLEN system [12]. The second generation of such a language is *knowledge query language* (KQL) which is being implemented in the VINLEN system. One way of interpreting knowledge scouts is that they are intelligent agents operating inside the inductive database, whose purpose is to seek requested relevant knowledge.

Through knowledge scouts or a graphical interface, the users of VINLEN have access to numerous operators for data and knowledge manipulation, knowledge creation, and knowledge application. These operators are grouped in categories such as *learn description*, *improve rules*, *learn trees*, *create clusters*, *apply statistics*, *generate equations*, and *determine optima*. Each of the categories consists of multiple operators, as shown in Figure 8. Because of the large variety of different operators and forms in which they generate knowledge, VINLEN is oriented so that the results of one operator are understood by other, applicable, operators. This enables the creation of knowledge scouts in which operators are applied to the results of applying other operators.

At the time that this chapter is being written, the VINLEN system is still under development, and new operators and functionalities are being added. This work follows the original design outlined by Ryszard Michalski.

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

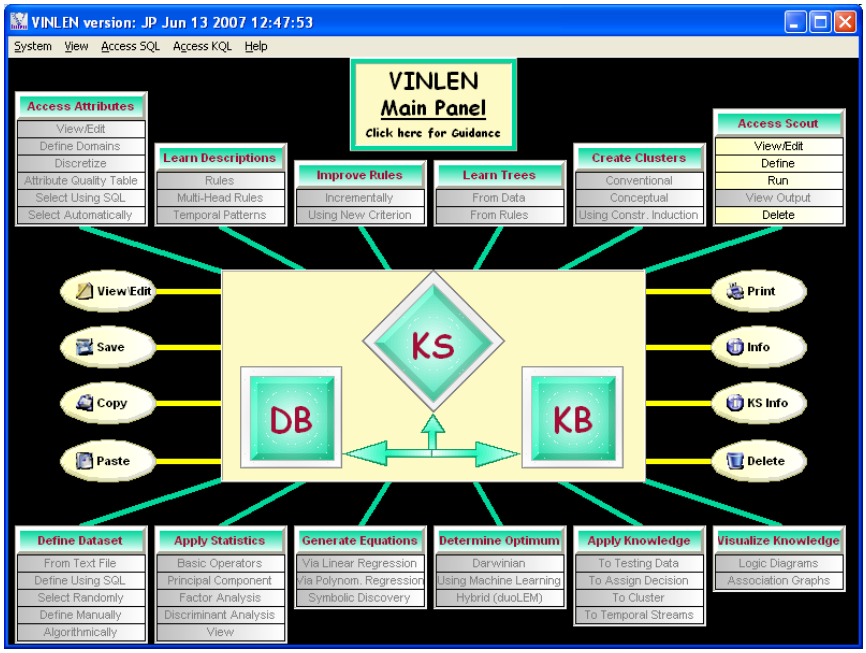



**Fig. 8.** The VINLEN system graphical interface.

## 7 Theoretical Aspects of Machine Learning

With his Theory and Methodology of Inductive Learning [22], Michalski defined several axes on which to categorize the inductive process, including input type (classified examples or unclassified observations), data types, whether or not counterexamples were presented, the types of covers learned and their relation to each other, and the use of original or constructed attributes. In this framework, he utilized VL1-style languages to express learned knowledge with high comprehensibility, which could be employed in concept learning, conceptual clustering, qualitative prediction, integrated qualitative and quantitative discovery, etc.

But more than that, he was fascinated by how we learn, for only through such an understanding could machines be made to truly emulate human learning process. He collaborated with numerous cognitive scientists, and among the fruits of this collaboration were the logic and theory of *plausible seasoning* [2], and *the inferential theory of learning* [23].

To utilize both of these ideas, Michalski noticed that we store many concepts in hierarchies. If an entity about which you are trying to extrapolate knowledge is a close relative (e.g., they share a common parent or grandparent in the hierarchy) of an entity whose characteristics you know, it stands to reason *plausibly* that the unknown

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com

entity may share characteristics similar to those of the known entity. Minnesota is near Iowa. They grow lots of corn in Iowa. Maybe they grow lots of corn in Minnesota too. Conversely, Arizona is far from Iowa and has a hotter, drier climate. They probably don't grow much corn there.

Thus, similization and dissimilization operators allow the induction of relationships between objects and concepts through the traversal of concept hierarchies. The inferential theory extended this idea by viewing all learning operators as implementers of *knowledge transmutations*, which would perform various tasks upon the network of hierarchies. Generalization would climb a hierarchy, while specialization would descend it. Abstraction would select only certain available information about an entity, while concretion would restore such information. Other transmutations would alter the hierarchies themselves and their interrelationships.

On these theoretical foundations, it was possible to explore the integration of diverse knowledge transmutations and their associated learning strategies into multistrategy systems. It is no coincidence that Ryszard Michalski was a prime impetus behind the first workshops, conferences and books on multistrategy learning.

## 8 Education

Given his belief in understandable knowledge representations, it is natural that Dr. Michalski also sought to educate all comers in the wider field of machine learning. Behind his impetus, the Machine Learning and Inference Laboratory produced a series of educational tools that could also serve as a springboard for research in the field.

In 1986, he entered into an agreement with representatives of the Boston Museum of Science to produce an introduction to and demonstration of inductive learning. The result was ILLIAN, a system that presented five different learning programs: AQ, INDUCE, CLUSTER, SPARC and ABACUS (see section 4). For each of these, the user was able to accept challenges on the types of problems these programs were capable of solving, and then create their own problems in the same toy domains and witness the programs' performance on them. They could also see brief descriptions of the learning algorithms used.

ILLIAN toured eight cities as part of the exhibition "Robots and Beyond: The Age of Intelligent Machines" before permanently taking root in Boston. Counters had been installed to track usage, and a report after the tour had concluded indicated that the program had been run over half a million times.

ILLIAN was oriented toward the general public, who may have had no AI exposure. In order to allow users go somewhat further into depth, an expanded version of the system, called EMERALD [9] doubled as a research tool. More complex problems were displayed and available to the user to create for the learning programs. EMERALD was displayed at classes in several countries, and distributed to hundreds of interested institutions. A recent reimplementation of one of EMERALD's modules, AQ, with a modern graphical interface is offered in the iAQ program [39][44], one of whose screens is presented in Figure 9.

In Advances in Machine Learning I, Koronacki, J., Ras, Z.W., Wierzchon, S.T. and Kacprzyk, J. (Eds.), 3-22, Springer-Verlag, 2010.
Original version available at spsringerlink.com


**Fig. 9.** A screen from the iAQ program.

## 9 Conclusion

In his research on machine learning, Ryszard S. Michalski contributed through new theories, algorithms, and organizational skills. He looked at problems differently than most researchers, which provided the field with his unique perspective, allowing him to originate several directions of research. Many of these directions, such as conceptual clustering, constructive induction, or natural induction, continue to evolve all over the world. His vision of the field has inspired many.

Perhaps the most important aspect of his research concerns the understandability of knowledge created by machine learning systems. While a majority of researchers in the field agree with this principle, very little has been actually done in this direction. Researchers construct more and more complicated models that slowly expand the barrier of predictive accuracy, and at the same time are often not understandable even by other researchers in the area. Dr. Michalski's ideas of natural induction or knowledge mining deserve further research in order to win back comprehensibility along with predictive accuracy.

This chapter briefly describes what we believe are some of the most important contributions of Dr. Michalski to machine learning research. It is, however, not

possible to give enough attention to many important aspects of work that appeared in over 350 publications. In the references section of this chapter, readers can find many important works, some of which are already considered classic in the field. Among the "conceptual offspring" worth mentioning, but not included in this chapter are: variable-precision logic, two-tiered concept representation, dynamic recognition, attributional ruletrees, dynamic interlaced hierarchies, and many others.

A comprehensive review of applications of natural induction and conceptual clustering is in a recent technical report [34]. Descriptions and lists of publications of the mentioned works and the many works of Dr. Michalski not mentioned here are maintained at the website of the GMU Machine Learning and Inference Laboratory, at http://www.mli.gmu.edu.

## References

[1] Bloedorn, E., Wnek, J., Michalski, R.S. and Kaufman, K.: AQ17 A Multistrategy Learning System The Method and Users Guide. Reports of the Machine Learning and Inference Laboratory, MLI 93-12, School of Information Technology and Engineering, George Mason University, Fairfax, VA, November (1993)

[2] Collins, A. and Michalski, R. S.: The Logic of Plausible Reasoning: A Core Theory. Cognitive Science, 13, pp. 1-49 (1989)

[3] Dietterich, T.G. and Michalski, R.S.: Discovering Patterns in Sequence of Events. Artificial Intelligence Journal, 25, No 2, pp. 187-232 (1985)

[4] Domanski, P. A., Yashar, D., Kaufman, K. and Michalski, R. S.: An Optimized Design of Finned-Tube Evaporators Using the Learnable Evolution Model. International Journal of Heating, Ventilating, Air-Conditioning and Refrigerating Research, 10, pp. 201-211 (2004)

[5] Falkenhainer, B. and Michalski, R.S.: Integrating Quantitative and Qualitative Discovery in the ABACUS System. In: Y. Kodratoff and R.S. Michalski (Eds.), Machine Learning: An Artificial Intelligence Approach, Vol. III, San Mateo, CA, pp. 153-190, Morgan Kaufmann Publishers, June (1990)

[6] Fürnkranz, J.: Separate-and-Conquer Rule Learning. Artificial Intelligence Review, 13, 1, pp. 3-54 (1999)

[7] International Intelligent Systems, Inc.: User's Guide to AURORA 2.0: A Discovery System. International Intelligent Systems, Inc. (1988)

[8] Kaufman, K.: INLEN: A Methodology and Integrated System for Knowledge Discovery in Databases. Ph.D. Dissertation, School of Information Technology and Engineering, Reports of the Machine Learning and Inference Laboratory, MLI 97-15, George Mason University, Fairfax, VA, November (1997)

[9] Kaufman, K. and Michalski, R.S.: EMERALD 2: An Integrated System of Machine Learning and Discovery Programs to Support Education and Experimental Research. Reports of the Machine Learning and Inference Laboratory, MLI 93-10, School of Information Technology and Engineering, George Mason University, Fairfax, VA, September (1993)

[10] Kaufman, K. and Michalski, R.S.: ISHED1: Applying the LEM Methodology to Heat Exchanger Design. Reports of the Machine Learning and Inference Laboratory, MLI 00-2, George Mason University, Fairfax, VA, (2000)

[11] Kaufman, K. and Michalski, R.S.: A Knowledge Scout for Discovering Medical Patterns: Methodology and System SCAMP. Proceedings of the Fourth International

Conference on Flexible Query Answering Systems, FQAS'2000, Warsaw, Poland, pp. 485-496, October 25-28 (2000)

[12] Kaufman, K. and Michalski, R.S.: The Development of the Inductive Database System VINLEN: A Review of Current Research. International Intelligent Information Processing and Web Mining Conference, Zakopane, Poland (2003)

[13] Kaufman, K. and Michalski, R.S.: From Data Mining to Knowledge Mining. In: Rao, C.R., Solka, J.L. and Wegman, E.J. (Eds.), Handbook in Statistics, Vol. 24: Data Mining and Data Visualization, pp. 47-75, Elsevier/North Holland (2005)

[14] Kaufman, K., Michalski, R.S. and Kerschberg, L.: Mining for Knowledge in Databases: Goals and General Description of the INLEN System. In: G. Piatetski-Shapiro and W. J. Frawley (Eds.), Knowledge Discovery in Databases, Menlo Park, CA , AAAI Press/The MIT Press (1991)

[15] Kaufman, K., Michalski, R.S., Pietrzykowski, J. and Wojtusiak, J.: An Integrated Multi-task Inductive Database VINLEN: Initial Implementation and Early Results: In *5th International Workshop on Knowledge Discovery in Inductive Databases, Revised Selected and Invited Papers*, Lecture Notes in Computer Science, 4747, 116-133, Springer (2007)

[16] Larson, J. and Michalski, R.S.: Inductive Inference of VL Decision Rules, Invited paper for the Workshop in Pattern-Directed Inference Systems, Hawaii, and published in SIGART Newsletter, ACM, No. 63, pp. 38-44, June 1977, May 23-27 (1977)

[17] Michalski, R.S.: On the Quasi-Minimal Solution of the General Covering Problem. Proceedings of the V International Symposium on Information Processing (FCIP 69)(Switching Circuits) , Vol. A3 , Yugoslavia, Bled, pp. 125-128, October 8-11 (1969)

[18] Michalski, R.S.: Synteza wyrazen minimalnych i rozpoznawanie symetrii funkcji logicznych. Prace Instytutu Automatyki PAN, Zeszyt 92, Warszawa, Instytut Automatyki Polskiej Akademii Nauk (1971)

[19] Michalski, R.S.: A Geometrical Model for the Synthesis of Interval Covers. Report No. 461, Department of Computer Science, University of Illinois, Urbana, June 24 (1971)

[20] Michalski, R.S.: A Variable-Valued Logic System as Applied to Picture Description and Recognition. Graphic Languages, In: F. Nake and A. Rosenfeld (Eds.), North-Holland Publishing Co. (1972)

[21] Michalski, R.S.: Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts. Journal of Policy Analysis and Information Systems, 4, 3, pp. 219-244, September (1980)

[22] Michalski, R.S.: A Theory and Methodology of Inductive Learning. In: R.S. Michalski, T.J. Carbonell and T.M. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach, pp. 83-134, TIOGA Publishing Co., Palo Alto (1983)

[23] Michalski, R.S.: Learning = Inferencing + Memorizing: Basic Concepts of Inferential Theory of Learning and Their Use for Classifying Learning Processes: In: Foundations of Knowledge Acquisition, Vol. 2: Machine Learning, pp. 1-41 (1993)

[24] Michalski, R.S., Learnable Evolution: Combining Symbolic and Evolutionary Learning. Proceedings of the Fourth International Workshop on Multistrategy Learning (MSL'98), Desenzano del Garda, Italy, pp. 14-20, June 11-13 (1998)

[25] Michalski, R.S.: LEARNABLE EVOLUTION MODEL Evolutionary Processes Guided by Machine Learning. Machine Learning, 38, pp.9-40 (2000)

[26] Michalski, R.S.: ATTRIBUTIONAL CALCULUS: A Logic and Representation Language for Natural Induction. Reports of the Machine Learning and Inference Laboratory, MLI 04-2, George Mason University, Fairfax, VA, April (2004)

[27] Michalski, R.S. and Baskin, A.B.: Integrating Multiple Knowledge Representations and Learning Capabilities in an Expert System: The ADVISE System. Proceedings of the 8th International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, pp. 256-258 (1983)

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

[29] Michalski, R.S., Carbonell, T.J. and Mitchell, T.M. (Eds.), Machine Learning: An Artificial Intelligence Approach, Palo Alto, TIOGA Publishing Co., Palo Alto (1983)

[30] Michalski, R.S., Carbonell, T.J. and Mitchell, T.M. (Eds.): Machine Learning: An Artificial Intelligence Approach, Vol. II, Los Altos, CA, Morgan Kaufmann Publishers, Inc. (1986)

[31] Michalski, R.S. and Kaufman, K.: INTELLIGENT EVOLUTIONARY DESIGN: A New Approach to Optimizing Complex Engineering Systems and its Application to Designing Heat Exchangers. International Journal of Intelligent Systems, 21, 12 (2006)

[32] Michalski, R.S. and Kaufman, K.: The AQ19 System for Machine Learning and Pattern Discovery: A General Description and User's Guide. Reports of the Machine Learning and Inference Laboratory, MLI 01-2, George Mason University, Fairfax, VA (2001)

[33] Michalski, R.S. and Kaufman, K.: Learning Patterns in Noisy Data: The AQ Approach. In: G. Paliouras, V. Karkaletsis and C. Spyropoulos (Eds.), Machine Learning and its Applications, pp. 22-38, Springer-Verlag (2001)

[34] Michalski, R.S., Kaufman, K., Pietrzykowski, J., Wojtusiak, J., Mitchell, S. and Seeman, W.D.: Natural Induction and Conceptual Clustering: A Review of Applications. Reports of the Machine Learning and Inference Laboratory, MLI 06-3, George Mason University, Fairfax, VA, June (2006)

[35] Michalski, R.S., Ko, H. and Chen, K.: SPARC/E(V.2), An Eleusis Rule Generator and Game Player. Reports of the Intelligent Systems Group, ISG 85-11, UIUCDCS-F-85-941, Department of Computer Science, University of Illinois, February (1985)

[36] Michalski, R.S., Ko, H. and Chen, K., Qualitative Prediction: The SPARC/G Methodology for Inductively Describing and Predicting Discrete Processes. In Expert Systems, Academic Press Inc., London (1986)

[37] Michalski, R.S. and Larson, J.: AQVAL/1 (AQ7) User's Guide and Program Description. Report No. 731, Department of Computer Science, University of Illinois, Urbana, June (1975)

[38] Michalski, R.S. and Larson, J.: Incremental Generation of VL1 Hypotheses: The Underlying Methodology and the Description of Program AQ11. Reports of the Intelligent Systems Group, ISG 83-5, UIUCDCS-F-83-905, Department of Computer Science, University of Illinois, Urbana, January (1983)

[39] Michalski, R.S. and Pietrzykowski, J.: iAQ: A program that discovers rules. AAAI-07 AI Video Competition, Twenty-Second Conference on Artificial Intelligence (AAAI-07), Vancouver, British Columbia, July 22–26 (2007)

[40] Michalski, R.S. and Stepp, R.: Learning from Observation: Conceptual Clustering. In: R.S. Michalski, T.J. Carbonell and T.M. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach, pp. 331-363, TIOGA Publishing Co., Palo Alto (1983)

[41] Michalski, R.S., Stepp, R. and Diday, E.: A Recent Advance in Data Analysis: Clustering Objects into Classes Characterized by Conjunctive Concepts. In: L. Kanal and A. Rosenfeld (Eds.), Progress in Pattern Recognition, Vol. 1, pp. 33-55, North-Holland (1981)

[42] Michalski, R.S. and Wojtusiak, J.: Reasoning with Meta-values in AQ Learning. Reports of the Machine Learning and Inference Laboratory, MLI 05-1, George Mason University, Fairfax, VA, June (2005)

[43] Michalski, R.S. and Wojtusiak, J.: Generalizing Data in Natural Language. Proceedings of the International Conference Rough Sets and Emerging Intelligent Systems Paradigms, RSEISP'07, Lecture Notes in Computer Science, Springer (2007)

[44] Pietrzykowski, J.: Demonstration and Application of Rule Discovery Methods Using iAQ. Workshop on Building Computational Intelligence and Machine Learning Virtual Organizations, George Mason University, Fairfax, VA, pp. 39-44, October 24 (2008)

[45] Seeman, W.D. and Michalski, R.S.: The CLUSTER3 System for Goal-oriented Conceptual Clustering: Method and Preliminary Results. Proceedings of The Data Mining and Information Engineering 2006 Conference, Prague, Czech Republic, July 11-13 (2006)

[46] Spackman, K.A., QUIN: Integration of Inferential Operators within a Relational Database. Reports of the Intelligent Systems Group, ISG 83-13, UIUCDCS-F-83-917, M.S. Thesis, Department of Computer Science, University of Illinois, Urbana (1983)

[47] Wnek, J., Kaufman, K., Bloedorn, E. and Michalski, R. S.: Inductive Learning System AQ15c: The Method and User's Guide. Reports of the Machine Learning and Inference Laboratory, MLI 95-4, George Mason University, Fairfax, VA, March (1995)

[48] Wojtusiak, J.: AQ21 User's Guide. Reports of the Machine Learning and Inference Laboratory, MLI 04-3, George Mason University, Fairfax, VA, September, (2004)

[49] Wojtusiak, J.: Handling Constrained Optimization Problems and Using Constructive Induction to Improve Representation Spaces in Learnable Evolution Model. Ph.D. Dissertation, College of Science, Reports of the Machine Learning and Inference Laboratory, MLI 07-3, George Mason University, Fairfax, VA, November (2007)

[50] Wojtusiak, J.: Data-driven Constructive Induction in the Learnable Evolution Model. Proceedings of the 16th International Conference Intelligent Information Systems, Zakopane, Poland, June 16-18 (2008)

[51] Wojtusiak, J. The LEM3 System for Multitype Evolutionary Optimization. Computing and Informatics, accepted for publication (2009)

[52] Wojtusiak, J. and Michalski, R.S.: The LEM3 Implementation of Learnable Evolution Model and Its Testing on Complex Function Optimization Problems. Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2006, Seattle, WA, July 8-12 (2006)

[53] Wojtusiak, J., Michalski, R.S., Kaufman, K. and Pietrzykowski, J.: The AQ21 Natural Induction Program for Pattern Discovery: Initial Version and its Novel Features. Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence, Washington D.C., November 13-15 (2006)