Computer Science • 18(3) 2017 http://dx.doi.org/10.7494/csci.2017.18.3.2364

Dorota Wilk-Kołodziejczyk

REASONING ALGORITHM FOR CREATIVE DECISION SUPPORT SYSTEM INTEGRATING INFERENCE AND MACHINE LEARNING

Abstract *In this paper, a reasoning algorithm for a creative decision support system is proposed. It allows for the integration of inference and machine learning algorithms. The execution of the learning algorithm is automatic since it is formalized by applying a complex inference rule, which generates intrinsically new knowledge using the facts already stored in the knowledge base as training data. This new knowledge may be used in the same inference chain to derive a decision. A solution of this type makes the reasoning process more creative and also allows for its continuation in cases when the knowledge base does not have the appropriate explicitly encoded knowledge. In this paper, an appropriate knowledge representation and inference model are proposed. Experimental verication is performed on the decision support system operating in a casting domain.*

Keywords reasoning algorithm, inferential theory of learning, decision support, rule induction, logic of plausible reasoning

Citation Computer Science 18(3) 2017: 317–338

1. Introduction

Classical reasoning algorithms applied in artificial intelligence (AI) offer a method of applying stored knowledge without providing new knowledge. This is the case of classical logic [16] rule-based systems using classical logic [18, 24], fuzzy logic [31], and Bayesian Networks – [21]). On the other hand, machine learning techniques may be creative and provide some diversity but are not integrated with the inference process. In this paper, an inference algorithm integrating these two approaches is proposed.

A solution of this type allows for the re-conceptualization of agent experiences depending on the context. As a result, reasoning may be performed even in cases not covered by the knowledge base. Some attempts have already been made to solve this problem [8, 14]); but so far, inference and execution of learning have not been integrated in the form of a deductive system such as the one shown in this study.

Using the proposed system, inference and machine learning can be integrated for various knowledge representations and reasoning techniques. The condition is to develop an inference process that could be formalized as a deductive system. For this purpose, both reasoning and learning assume the application of inference (proof) rules. The results of the machine learning algorithm should take a form that may be represented in the chosen formalism, thus allowing for the further use of these results in the inference process. For example, rule induction may generate rules that can be used in a rule-based system.

In our solution, we assume that the knowledge representation has two types of inference rules: simple inference rules (like Modus Ponens) and complex inference rules, the application of which corresponds to the execution of a learning algorithm. Complex rules are used in an inference chain if the reasoning process is not able to continue classical reasoning. Training data relies on facts already stored in the knowledge base. The new knowledge may be used in the same inference chain to derive a decision.

The solution proposed is formulated as a Multi-strategy Inference and Learning System (MILS). The idea is based on the Inferential Theory of Learning disclosed in [19], where learning and inference can be presented as a goal-guided exploration of the knowledge space using operators called knowledge transmutations.

In our research, we have chosen a complex knowledge representation and reasoning formalism called the Logic of Plausible Reasoning (LPR) [7].

As a result, the developed algorithm allows us to combine several knowledge manipulation techniques during the process of reasoning. To infer a decision, there are options to use background knowledge, simple proof rules (such as Modus Ponens or generalization), or complex ones (machine learning or searching algorithms).

In the following sections, the related research is discussed, and the MILS model and inference algorithm are presented. Next, LPR basics and the related software are described. The results of the experiments to support the material choice for casting (knowledge base, reasoning scenarios, and performance tests) conclude the work.

2. Related research

The Inferential Theory of Learning (ITL), on which the MILS is based, was formulated by Ryszard Michalski [19]. Michalski *et al.* also developed ITL partial implementation – which is an INTERLACE system [2], wherein knowledge representation is based on Dynamically Interlaced Hierarchies (DIH see below). The system can generate sequences of knowledge operations that will enable the derivation of a target trace from the input hierarchies and traces. Machine learning was not integrated in this system.

DIH formalism was proposed by Michalski *et al.* [12, 13]. Knowledge consists of a static part represented by hierarchies and a dynamic part, which are traces representing the relationships between hierarchy nodes. The DIH distinguishes three types of hierarchies: types, components, and priorities. The latter type of hierarchy can be divided into subclasses: hierarchies of measures (used to represent the physical quantities), hierarchies of quantification (allowing quantifiers to be included in traces, such as one, most, or all, for example), and hierarchies of schemes (used as a means for defining multi-argument relationships and needed to interpret the traces).

Logic of Plausible Reasoning (LPR), which is the base of DIH works, was proposed by Collins and Michalski [7]. The aim of the study was to identify the reasoning patterns used by humans and create a formal system, which would be able to represent these patterns. The mere objective set by the creators has made the LPR signicantly dierent from other known knowledge representation methods, such as classical logic, fuzzy logic, multi-valued logic, Demster-Shafer theory, probabilistic logic, Bayesian networks, semantic networks, ontologies, rough sets, or default logic. There are many inference rules in the LPR that are not present in the formalisms mentioned above. Equally important is also the fact that there are many parameters specied to represent the uncertainty of knowledge. The basic operations performed on the knowledge dened in the LPR include:

- **abduction and deduction** – used to explain and predict the characteristics of objects based on domain knowledge;
- **generalization and specialization** – allow for the generalization and refining of information by changing the set of objects to which this information relates to a set larger or smaller;
- **abstraction and concretization** – change the level of detail in the description of objects;
- **similarity and contrast** – allow inference by analogy or lack of similarity between objects.

The experimental results confirming that the methods of reasoning used by humans can be represented in the LPR are presented in subsequent papers [4, 5].

Integration of expert systems and machine learning was analyzed some time ago. A system presented in [14] is based on neural logic networks corresponding to

three-valued logic. The system allows for the adaptive learning of new rules from its experience [1].

In [8] a neural network was also applied to overcome the brittleness of classical expert systems. It is used for choosing the most-appropriate questions for the current case. The description of user interaction with the system is collected as training data for the network.

In [10] an adaptive expert system is proposed for aircraft maintenance. It recommends the most-accurate action for symptoms reported by a user. Like in the examples above, learning uses historical data (in this case, a repair register) to update association weights between symptoms and actions. The certainty of the suggested diagnosis increases in the case of successful prediction or decreased in the case of failure. Symptoms may be also combined using generalization.

In the solutions presented above, machine learning algorithms are not part of the formal reasoning system. Therefore, the integration of machine learning and reasoning is not complete.

In the CoMES system [3], an attempt has been made to combine several popular techniques from the fields of Artificial Intelligence and Software Engineering. Machine learning is used to update the knowledge base, which can be accessed by a few algorithms in parallel. The system uses agent architecture to integrate knowledge from human experts and other expert systems.

3. MILS model

The reasoning algorithm is based on the MILS model, which assumes that the knowledge representation and reasoning method can be formalized as a labeled deductive system (LDS) [9]. Knowledge is represented by formulas F, which may be uncertain. To model uncertainty, a *label algebra* may be used:

$$\mathcal{A} = (A, \{f_{r_i}\}). \tag{1}$$

where A is a set of labels that estimate the uncertainty of formulas.

Labeled formula is a pair $f : l$ where: $f \in F$ is a formula and $l \in A$ is a label. A finite set of labeled formulas can be considered as a knowledge base. Functions f_{r_i} are used to calculate labels of reasoning results.

The inference patterns should be defined as *proof rules*. Each rule r_i has a sequence of premises α_i (of length n_i) and a conclusion α:

$$\frac{\begin{array}{l}\alpha_1 : l_1 \\ \alpha_2 : l_2 \\ \vdots \\ \alpha_n : l_n\end{array}}{\alpha : l} \tag{2}$$

For rule r_i, the plausible label of its conclusion is calculated using $f_{r_i} : A^{n_i} \rightarrow A$, hence $l = f_{r_i}(l_1, ..., l_n)$.

Having introduced the inference rules, one can define the proof of labeled formula φ selected from a set of labeled formulas comprised in knowledge base (KB). This proof can be dened as a tree. The tree P is a proof of labeled formula $\varphi : l$ selected from a set of labeled formulas comprised in knowledge base KB, if a root node of P is equal to $\varphi : l$ and for every node $\psi : l_\psi$:

- if $\psi : l_\psi$ is a leaf, then $\psi : l_\psi \in KB$,
- else, there are nodes $(\psi_1 : l_{\psi_1}, ..., \psi_k : l_{\psi_k})$, connected to $\psi : l_\psi$ and a proof rule r_i such, that $\psi : l_\psi$ is a consequence of r_i and $(\psi_1 : l_{\psi_1}, ..., \psi_k : l_{\psi_k})$ are its premises (label of ψ is calculated using f_{r_i}).

The MILS adds additional and more-general proof rules to the basic deductive system. They are called knowledge transmutations. As a result, we have three types of these:

- simple, based on syntactic operations (e.g., Modus Ponens),
- complex, based on procedure execution (e.g., rule induction algorithms, clustering methods,
- search (database or web searching procedures).

Knowledge transmutation can be represented as a triple:

$$kt = (p, c, a), \tag{3}$$

where: p is a (possibly empty) premise or preconditions, c is a consequence (pattern of formula(s) that can be generated), and a is an action (empty for simple transmutations) that should be executed to generate consequence if premises are true according to the knowledge base.

Every transmutation has its cost assigned. The cost should represent the computational complexity and/or other important resources that are consumed (e.g., database access or search engine fees). Usually, simple transmutations result in lower costs and complex ones result in higher costs [25].

4. Reasoning algorithm

According to [28], the MILS inference algorithm (see algorithm 1) is an adaptation of the LPR proof algorithm, where proof rules are replaced by more-general knowledge transmutations. It is based on the AUTOLOGIC system developed by Morgan [20]. To limit the number of nodes and to generate optimal inference chains, algorithm A* [11] is used.

The input data is a set of labeled formulas stored in knowledge base KB and a hypothesis (question) represented by formula φ, which should be derived from KB. If label $l \in A$ exists such that $\varphi : l$ can be inferred from KB, the appropriate inference chain is returned; otherwise, the procedure exits with failure.

Algorithm 1 Reasoning algorithm for MILS.

Input: φ – formula, KB – finite set of labeled formulas
Output: If $\exists l \in A$ such that $\varphi : l$ can be inferred from KB: success, P – inference chain of $\varphi : l$ from KB; else: failure

1: T := tree with one node (root) $s = [\varphi]$
2: $OPEN := [s]$
3: **while** $OPEN$ is not empty **do**
4: n := the first element from $OPEN$
5: Remove n from $OPEN$
6: **if** $n = []$ **then**
7: Generate proof P using path from s to n
8: Exit with success
9: **else if** the first formula of n represents action **then**
10: Execute action
11: **if** action was successful **then**
12: add action's results to KB
13: E:=nodes generated by removing from n action formula
14: **end if**
15: **else**
16: K := knowledge transmutations, whose consequence can be unified with the first formula of n
17: E := nodes generated by replacing the first formula of n by premises and action of transmutations from K and applying substitutions from unifier generated in the previous step
18: **if** the first formula from n can be unified with element of KB **then**
19: Add to E node obtained from n by removing the first formula and applying substitutions from the unifier
20: **end if**
21: **end if**
22: Remove from E nodes generating loops
23: Append E to T connecting nodes to n
24: Insert nodes from E to $OPEN$
25: Order $OPEN$
26: **end while**
27: Exit with failure

Agent experience and the context description should be also stored in KB as LPR formulas.

This algorithm generates tree T, the nodes (N) of which are labeled by sequences of formulas. Every edge of T is labeled by a knowledge transmutation, whose consequence can be unified with the first formula of a parent node or is labeled by term $kb(l)$ if the first formula of a parent node can be unified with $\psi : l \in KB$. s is the root of T, and it is labeled by $[\varphi]$. The goal is to generate a node labeled by an empty set of formulas.

As previously mentioned, the A* algorithm is used to limit the number of expanded nodes. Therefore, nodes in the $OPEN$ sequence are ordered according to the values of evaluation function $f : N \to R$, which is defined as follows:

$$f(n) = g(n) + h(n), \tag{4}$$

where: $g : N \to R$ represents the actual cost of the inference chain, using knowledge transmutation costs, and label of φ that can be generated, and $h : N \to R$ is a heuristic function which estimates the cost of the path from n to the goal node (e.g., minimal knowledge transmutation cost multiplied by the length of n can be used).

All formulas in the proof path can be forgotten when a new task is executed. But, it is also possible to keep learned formulas in the cache knowledge base together with a counter that will indicate the number of proofs in which this formula is used. Using a formula in some proofs should increase the number of counted events, whereas a lack of use should decrease this number. If the number of counted events is equal to 0, the formula should be removed from the temporal knowledge base [15].

5. LIIS system

In this section, the LPR Intelligent Information System (LIIS) used in experiments is described. Knowledge is represented with the logic of plausible reasoning (LPR); therefore, this formalism is introduced at the very beginning. As a next step, the main features and implementation details of the system are described. Finally, the label algebra is presented.

5.1. Introduction to LPR

To show that the MILS may be applied to complex inference systems, LPR has been chosen for basic knowledge representation and reasoning. Instead of LPR, another technique that can be formulated using LDS may be used if needed.

The language used by LPR consists of a countable set of constants C, variables X, the seven relational symbols, and logical connectives $\to$ and $\wedge$. Formally, it is a quadruple: $L = (C, X, \{V, H, B, E, S, P, N\}, \{\to, \wedge\})$. Relational symbols (V, H, B, E, S, P, N) are used for defining the following relationships:

- H defines the hierarchy between concepts; notation $H(o_1, o, c)$ means that o_1 is o in context c;
- B represents the fact that one object is placed below another in a hierarchy;
- V represents statements: notation $V(o, a, v)$ represents the fact that object o has an attribute a equal to v;
- E represents relationships; notation $E(o_1, a_1, o_2, a_2)$ means that the values of attribute a_1 of first object o_1 depend on the values of attribute a_2 of second object o_2;
- S determines the similarity between objects; notation $S(o_1, o_2, c)$ represents the fact that o_1 is similar to o_2 in context c;

- P represents the order between concepts: notation $P(o_1, o_2)$ means that concept o_1 precedes concept o_2;
- N compares the concepts; notation $N(o_1, o_2)$ means that concept o_1 is different from concept o_2. This relationship does not appear in the knowledge base, but it is only used as a premise for some implications.

To represent the vagueness of the knowledge, it is possible to extend the statement definition and allow it to use composite value $[v_1, v_2, \ldots, v_n]$ (list of elements of C). It can be interpreted that object o has an attribute a equal to v_1 or v_2, ..., or v_n. If $n = 1$ instead of $V(o, a, [v_1])$, notation $V(o, a, v_1)$ is used. In the statements, a value should be placed in a hierarchy below the attribute: if $V(o, a, [v_1, v_2, \ldots, v_n])$ is in a knowledge base, there should also be $H(v_i, a, c)$ for any $1 \leqslant i \leqslant n, c \in C$.

An LPR formula is any atomic formula: $H(o_1, o_2, c), B(o_1, o_2), V(o, a, v)$, $E(o_1, a_1, o_2, a_2), S(o_1, o_2, c), P(o_1, o_2)$, where $o, o_1, o_2, a, a_1, a_2, c, v \in C$, a conjunction of atomic formulas and implications in the form of $\alpha_1 \wedge \alpha_2 \wedge \ldots \alpha_n \rightarrow V(o^\alpha, a^\alpha, v^\alpha)$, where $n \in \mathrm{N}, n > 0$. It is assumed that α_i has the form of $V(o_i^\alpha, a_i^\alpha, v_i^\alpha)$, $P(v_i^\alpha, w_i^\alpha)$ or $N(v_i^\alpha, w_i^\alpha)$, and $o^\alpha, o_i^\alpha, a^\alpha, a_i^\alpha, v^\alpha, v_i^\alpha, w_i^\alpha \in C \cup X$ for $1 \leqslant i \leqslant n$.

The most-commonly-used proof rules operate on the statement (others can be found in [7]). An index attached to the name of the rule tells us what is being transformed: o is an object, and v is the value. These rules are shown in Table 1.

Table 1

Rules transforming object-attribute-value triples.

GEN_o	$\dfrac{(o_1, o, c)\quad (o, a, o, c)\quad (o_1, a, v)}{(o, a, v)}$	$SPEC_o$	$\dfrac{(o_1, o, c)\quad (o, a, o, c)\quad (o, a, v)}{(o_1, a, v)}$	SIM_o	$\dfrac{(o_1, o_2, c)\quad (o_1, a, o_1, c)\quad (o_2, a, v)}{(o_1, a, v)}$
GEN_v	$\dfrac{(v_1, v, c)\quad (a, o, a, c)\quad (o_1, o, c_2)\quad (v, a)\quad (o_1, a, v_1)}{(o_1, a, v)}$	$SPEC_v$	$\dfrac{(v_1, v, c)\quad (a, o, a, c)\quad (o_1, o, c_1)\quad (o_1, a, v)}{(o_1, a, v_1)}$	SIM_v	$\dfrac{(v_1, v_2, c)\quad (a, o, a, c)\quad (o_1, o, c_1)\quad (v_1, a)\quad (o_1, a, v_2)}{(o_1, a, v_1)}$
		MP	$\dfrac{\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow (o, a, v)\quad \alpha_1\quad \vdots\quad \alpha_n}{(o, a, v)}$		

GEN_o and $SPEC_o$ are the generalization and specialization of the objects in the statements, respectively, while GEN_o and $SPEC_o$ are similar transformations of the values, SIM_o represents reasoning by analogy (similarity) between objects, while SIM_v represents the analogy of the values. MP is the classical Modus Ponens inference rule.

5.1.1. Reasoning

The main function of the system is to perform inference based on the knowledge base available and user-defined hypothesis (also known as a query). This function is based on the MILS framework.

The query is a statement that can include constants, variables, and numeric values. The hypothesis is verified by the inference engine. If it can be proven to have a non-zero probability, a proof is submitted. If there are several proofs, all of them are presented to the user along with the relevant information concerning their credibility.

The user can choose the maximum depth of the proof tree as well as the knowledge transmutations that are to be used in the process.

5.1.2. Expert system

The system provides an engine to perform decision support. This functionality has been extensively used in the case study described in the next section. Expert system scenarios use both formulas from the base and information provided by a user executing the decision support procedure. The user can supply knowledge by selecting an answer from the list or filling in inputs with numerical values. Questions can be skipped. Scenarios can be developed using a GUI.

5.1.3. Search

The search is understood as a search in the knowledge base for the concept that has attributes of the given values. This operation can be performed in two modes: standard and rapid (the latter uses fewer possibilities offered by LPR). As in the case of inference, here we can also determine the maximum depth of the proof tree. In addition, we can select the hierarchy (or a sub-tree of the hierarchy) in which the system is supposed to perform the search.

5.1.4. Knowledge base edition

Application supports the edition of knowledge base elements, providing tools that facilitate this process. A single formula is created with a form, suggesting object names based on the ones already used in the system. Formulas of different types are placed in respective tables, where they can be ltered by the name of one of their objects [30].

5.1.5. Machine learning

In the knowledge base, machine learning can be carried out manually. The collected knowledge (statements or results of reasoning) forms the training and test data sets. Within the given configuration, the system allows us to define the range of knowledge for learning, choose the learning algorithm, and set all necessary parameters. Before adding the learned rules to the knowledge base, the user may review and verify the new knowledge [22].

Machine learning algorithms may be automatically executed during the inference process. Currently, two complex knowledge transmutations are defined. Both apply the rule induction using the AQ [17] and C4.5 [23] algorithms. Consequence c (see (3)) has the form of a statement for both transmutations. To reduce the computation time, the user may limit the learning process to a set of attributes called the *category set*. In premise p, it is checked if it is possible to generate enough examples from the knowledge base. Examples are divided into training and testing data. The second set is used to estimate the strength of the rules learned.

5.2. Description of LIIS implementation

In the description of the developed application, an attempt was made to characterize the solution, heading for the widest-possible use of the existing development tools while providing the functionality needed to effectively meet the functional requirements. As a result, the LPR Intelligent Information System is a web application created with Google Web Toolkit, a solution supporting the development of browser-based applications. The technology affects the system architecture, dividing it into three logical parts. The LIIS architecture is shown in Figure 1.

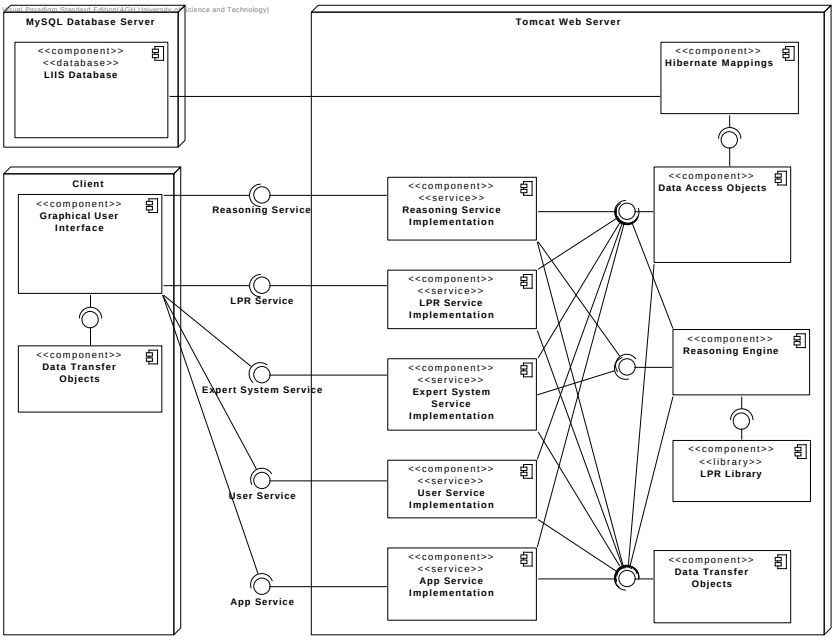

Figure 1. LIIS architecture

The server-side part of the system is responsible for the realization of the main features. Using LPR-Library, the reasoning engine implements inferencing, executes machine learning, and conducts expertise. The engine is used by the services responsible for providing these functionalities to users: Reasoning Service and Expert System

Service. The LPR Service allows storing and obtaining knowledge base elements from the MySQL database with the use of Data Access Objects. The persistence layer takes advantage of Hibernate object-relational mapping and stores formulas, expert system scenarios, user data, and knowledge-based metadata. System management is provided via User Service and App Service [6].

The client part of the application contains JavaScript views compiled from Java classes. It communicates with the server-side part through Remote Proxy Calls, where data is transported via HTTP as Data Transfer Objects - plain serializable Java classes shared by both sides. LIIS is built with the Maven Dependencies Management Tool. It runs on the Apache Tomcat web server.

In the current implementation of the reasoning engine, the following formulas can be used: statements (V), hierarchy (H), similarity (S), and implications P and N. The applied transformations include object rules and value generalization, specialization, similarity, Modus Ponens, ordering, and hierarchy transitivity.

5.3. Label algebra

In our studies, we have used a simplied version of label algebra dened in [27]. To represent certainty, the system uses the following coefficients (which are real numbers within a range of [0,1]):

- for formulas V – confidence;
- for formulas S – similarity rate;
- for formulas H – typicality and dominance;
- for formulas B – confidence;
- for formulas I (implications) – strength.

Formulas P and N are certain (have label equal to 1.0).

The certainty label of the statement (which is the conclusion of proof rule r_i) is a product of the label of each of the following premises:

$$f_{r_i}(l_1, l_2, \ldots, l_{n_i}) = \prod_{i=1}^{n_i} l_i \tag{5}$$

If the premise represents the hierarchy, it is typicality used in object transformation, and dominance is used in value transformation.

6. Experimental results: material choice support system

The MILS framework was tested before in two small domains [29, 26]. To show the advantages of the proposed solution on a larger scale, a decision support system was developed in the domain that is complex enough, contains hierarchies of the objects, and is characterized by a number of parameters of an intuitive nature, dicult to measure. The system supports the choice of metal-product-manufacturing technology (casting technology included). The knowledge base consists of more than 700 formulas.

Often, the choice of technology for the manufacture of metal items and of the material from which this item is to be produced stems from the experience and knowledge of the engineer designing this item. These human aspects are dicult to represent using formal languages. When the task of designing machine parts is undertaken, the parameters that the item should have and the related operational and utility functions must be taken into account. This also applies to the case of material conversion. A new type of material must provide at least the same mechanical properties and reliability as the original one. The choice of the manufacturing process is aected by batch size, dimensional accuracy, overall dimensions, complexity, type of the necessary machining, heat treatment, etc. All of these factors also create costs. In this situation, the problem that the designer of a particular product (machine part) has to face and solve consists of selecting the material as well as the technology of its manufacture to ensure that the specic technical requirements are satised while allowing for the maximum reduction of production costs. In the LIIS embodiment considered here, it is very important to indicate the appropriate material that can replace the traditional materials (forged steel, cast steel). This material can be Austempered Ductile Iron (ADI), which has a favorable relationship between tensile strength (Rm) and elongation (A), oering signicantly lower manufacturing costs (savings of approx. 20%) at the same time. However, the decision about the possible use of ADI must be based on a more-detailed analysis of the requirements imposed on a particular product and its characteristics (to mention as an example, the damping capacity, corrosion resistance, dimensions, batch size, and weight of a single item).

It was assumed (including the technology used) that the ADI items would be made by casting with further heat treatment and machining, while items made of carburized steel (e.g., 16MnCr5) would be cut from sheet metal, rough machined, carburized, and then nished. Therefore, comparing these two materials, the following parameters were taken into account: the cost of application, heat treatment, machining, cutting, pattern equipment, molding technology, melting of metal and pouring of molds, and price of the carburized steel sheet.

It is understood that the low-volume production includes up to 50 pieces of castings weighing between 0 and 25 kg. Low-volume production is also comprised of up to ten cast pieces (if the casting weight is 25–500 kg). If the casting weight exceeds 500 kg, the low-volume production consists of one cast piece. Medium-volume production covers 50–5000 pieces for a weight range between 0 and 25 kg, 10–100 pieces for a weight range of 25–500 kg, and 2–10 pieces for a total weight of more than 500 kg. All values above the last level stand for large-lot production.

The batch size (production volume) depends on the weight of the products in each of the three type ranges. This helps us better understand the comparison of prices for the same product made from ADI and carburized steel for different batch sizes and product weights.

The knowledge base shown below (expressed with LPR formalisms) was constructed basing on data prepared with the significant participation of technologists,

who helped in the proper interpretation of technical requirements and resolved doubts arising when some of the equations were formulated.

The core of the knowledge base consists of hierarchies. They were defined during consultations with experts. Formulas representing the hierarchy of ADI are presented below. They represent the facts that ADI is a kind of cast_iron and defines its 63 subtypes (ADI_GSJ-1400-1, ADI_1, ADI_2, ..., ADI_31, ..., ADI_34, ADI_41, ..., ADI_44, ADI_51, ..., ADI_68, ...). The context is related to cost, production volume, application, and mechanical properties. The first label value (typicality) is high (often equal to 1.0), which means that the certainty of the specialization of objects and values (SPECo and SPECv) will also be high. The second label (dominance) is low. Part of the hierarchy is presented below:

1. H(adi, cast_iron, cost): 0.8: 0.1
2. H(adi, cast_iron, volume_production):0.8:0.1
3. H(adi_gsj-1400-1,adi, application):1.0:0.1
4. H(adi_4, adi, application):1.0:0.1
5. H(adi_42, adi, application):1.0:0.1
6. H(adi_52, adi, application):1.0:0.1
7. H(adi_gsj-1400-1, adi, tensile_strength_Rm):1.0:0.1
8. H(adi_4, adi, tensile_strength_Rm):1.0:0.1
9. H(adi_42, adi, tensile_strength_Rm):1.0:0.1
10. H(adi_52, adi, tensile_strength_Rm):1.0:0.1
11. H(adi_gsj-1400-1, adi, minimal_elongation_A):1.0:0.1
12. H(adi_4, adi, minimal_elongation_A):1.0:0.1
13. H(adi_42, adi, minimal_elongation_A):1.0:0.1
14. H(adi_52, adi, minimal_elongation_A):1.0:0.1
15. H(adi_gsj-1400-1, adi, cost):1.0:1.0
16. H(adi_4, adi, cost):1.0:0.1
17. H(adi_42, adi, cost):1.0:0.1
18. H(adi_52, adi, cost):1.0:0.1

The following notation denotes the similarity between two types of materials (in this case, ADI and carburized steel) discussed in terms of their application. The label represents a high similarity level.

S(adi,steel_carburized,application):0.8

In the statements presented below, the minimum elongation and tensile strength of the selected steel grades are expressed. Labels representing certainty have high values. Similar statements have been prepared for other types of ADI (like ADI_4, ADI_42, ADI_52, etc.). Some parameters are not known, and their corresponding statements are missing.

1. V(adi, application, rake):1.0
2. V(adi_gsj-1400-1, minimal_elongation_A, 1):1.0
3. V(adi_gsj-1400-1, tensile_strength_Rm, 1400):1.0
4. V(engjs_14001, chemical_composition_c, 3.462-3.524):1.0
5. V(adi_gsj-1400-1, chemical_composition_si, 2.39-2.51):1.0
6. V(adi_gsj-1400-1, chemical_composition_mn, 0.334-0.422):1.0
7. V(adi_gsj-1400-1, chemical_composition_p, 0-0.146):1.0
8. V(adi_gsj-1400-1, chemical_composition_s, 0-0.0104):1.0
9. V(adi_gsj-1400-1, chemical_composition_mo, 0.286-inf):1.0

10. V(adi_gsj-1400-1, chemical_composition_ni, 1.296-1.598):1.0
11. V(adi_gsj-1400-1, chemical_composition_cu, 0.23-0.405):1.0
12. V(adi_gsj-1400-1, chemical_composition_mg, 0-0.0422):1.0
13. V(adi_gsj-1400-1, chemical_composition_ti, 0.0188-inf):1.0
14. V(adi_gsj-1400-1, chemical_composition_cr, 0-0.108):1.0
15. V(adi_gsj-1400-1, austenization_time, 105-inf):1.0
16. V(adi_gsj-1400-1, austenization_temp, 867.5-895):1.0
17. V(adi_gsj-1400-1, hardening_time, 187.5-inf):1.0
18. V(adi_gsj-1400-1, hardening_temp, 0-293.75):1.0

The remaining formulas have the form of implication. The first three allow us to recommend a material for production. They have conclusion V(casting, material_alternative, X). The more parameters that are checked (and the more premises has the rule), the more certain is the answer. The first implication checks application, costs, tensile strength, and minimum elongation, and it has certainty of 1.0. The fourth rule only checks application; therefore, its certainty is equal to 0.25. Other rules (5–22) allow us to predict the production costs at a selected value of the batch size and product weight.

1. V(casting,application_required,A) ∧ V(X, application,A) ∧ V(casting,cost_required, COST_MAX) ∧ V(X,cost,COST_CALCULATED) ∧ P(COST_CALCULATED,COST_MAX) ∧ V(casting,tensile_strength_Rm_required,STRENGTH_MIN) ∧ V(X, tensile_strength_Rm,C) ∧ P(STRENGTH_MIN,C) ∧ V(casting,minimal_elongation_A_required,ELONG_MIN) ∧ V(X,minimal_elongation_A,E) ∧ P(ELONG_MIN,E)] → V(casting,material_alternative,X):1.0
2. V(casting, application_ required,A) ∧ V(X, application,A) ∧ V(casting,cost_required,COST_MAX) ∧ V(X,cost,COST_CALCULATED) ∧ P(COST_CALCULATED,COST_MAX) ∧ V(casting, tensile_strength_Rm_required, STRENGTH_MIN) ∧ V(X, tensile_strength_Rm,C) ∧ P(STRENGTH_MIN,C) → V(casting, material_alternative,X):0.75
3. V(casting, application_required,A) ∧ V(X, application,A) ∧ V(casting, cost_required,COST_MAX) ∧ V(X, cost, COST_CALCULATED) ∧ P(COST_CALCULATED, COST_MAX)] → V(casting, material_alternative,X):0.5
4. V(casting, application_required, A) ∧ V(X, application, A)] → V(casting, material_alternative,X):0.25
5. V(casting, weight, small) ∧ V(casting, volume_production, small) → V(adi,cost,70):0.8
6. V(casting, weight, small) ∧ V(casting, volume_production, medium)] → V(adi,cost,20):0.9
7. V(casting, weight, small) ∧ V(casting, volume_production, large)] → V(adi,cost,16):1.0
8. V(casting, weight,medium) ∧ V(casting, volume_production, small)] → V(adi,cost,21):0.8
9. V(casting, weight, medium) ∧ V(casting, volume_production, medium)] → V(adi,cost,18):0.9
10. V(casting, weight, medium) ∧ V(casting, volume_production, large)] → V(adi,cost,14):1.0
11. V(casting, weight, large) ∧ V(casting, volume_production, small)] → V(adi,cost,16):0.8
12. V(casting, weight,large) ∧ V(casting, volume_production, medium)] → V(adi,cost,14):0.9
13. V(casting, weight,large) ∧ V(casting, volume_production,large)] → V(adi,cost,12):1.0
14. V(casting, weight,small) ∧ V(casting, volume_production,small)] → V(steel_carburized,cost,23):0.8
15. V(casting, weight, small) ∧ V(casting, volume_production, medium)] → V(steel_carburized,cost,20):0.9
16. V(casting, weight, small) ∧ V(casting, volume_production,large)] → V(steel_carburized,cost,19):1.0
17. V(casting, weight,medium) ∧ V(casting, volume_production,small)] → V(steel_carburized,cost,21):0.8
18. V(casting, weight, medium) ∧ V(casting , volume_production, medium)] → V(steel_carburized,cost,21):0.9

19. V(casting, weight, medium) ∧ V(casting, volume_production,large)] →
    V(steel_carburized,cost,22):1.0
20. V(casting, weight,large) ∧ V(casting, volume_production,small)] →
    V(steel_carburized,cost,30):0.8
21. V(casting, weight,large) ∧ V(casting, volume_production, medium)] →
    V(steel_carburized,cost,40):0.9
22. V(casting, weight,large) ∧ V(casting, volume_production,large)] →
    V(steel_carburized,cost,50):1.0

7. Usage scenarios

In order to check the correctness of the inference algorithm, various scenarios have been developed. Below, the selected three scenarios are presented. The description of the scenarios contains the hypotheses, user responses, proof obtained, proof rules used during inference, and description of the inference process.

The goal of the decision support system is to find a material that, while fitting the requirements, will also reduce the casting cost through its use.

7.1. Scenario 1

The first scenario illustrates a simple case in which all knowledge necessary for reasoning is given explicitly in the knowledge base. The application of the material is a rack[1], the maximum cost limit is equal to 15, the product weight is heavy, the batch size is large, the minimum tensile strength Rm is equal to 1100, and the hardness is high. As a result, the system recommends ADI_4 with 1.0 confidence.

The proof was obtained by double application of the Modus Ponens (MP) rule and double object specialization (SPECo) rule. It is presented in Figure 2.

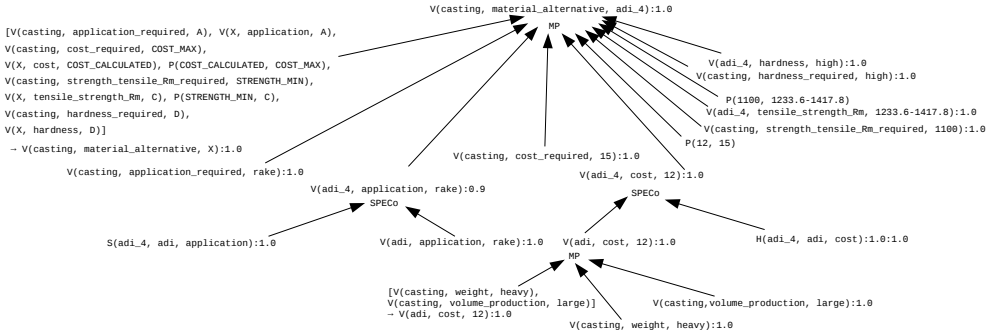

Figure 2. Graphical presentation of proof found in Scenario 1.

[1]A rack is a tool used in sewage-treatment plants. Its main task is to mix organic materials such as straw, grass, hay, etc. with the semi-liquid material obtained from municipal waste-water treatment after suitable processing. This is then mixed with soil and refining additives to obtain a mineral fertilizer used in agriculture.

In the first step, the MP rule was applied to Implication no. 1, which means that, if the required application of casting under consideration is equal to A (premise 1) and is the same as the application allowed for an alternative material in the rule marked by variable X (Premise 2), the required maximum cost is equal to COST_MAX (Premise 3), and the cost calculated for an alternative material is equal to COST CALCULATED (Premise 4) and is lower than the maximum cost (Premise 5), the required minimum tensile strength Rm is STRENGTH_MIN (Premise 6), and for an alternative material, it is C (Premise 7) and is higher than STRENGTH_MIN (Premise 8), and the required hardness described as HARDNESS (Premise 9) is the same as for alternative material (Premise 10), then the alternative material (X) should be used with 1.0 confidence.

Premises 1 and 3 are adaptable to the knowledge base elements or answers to questions. Premise 2 (application acceptable for ADI_4) was inferred using the SPECo object specialization rule because ADI_4 is a typical ADI in terms of application, and it is known that ADI may be used to produce racks. In a like manner, Premise 4 was derived using the SPECo specialization rule, and knowing that ADI_4 is a typical ADI in terms of the manufacturing cost, calculating this cost is based on the casting weight and using Implication no. 13 as above. Premises 5–10 can be unified with the knowledge base elements or answers to questions.

7.2. Scenario 2

User requirements in this scenario are the following:

- the application is also rack,
- the maximum allowable cost is 15,
- the casting weight (diameter) is medium this time,
- the batch size is large,
- the minimum tensile strength Rm is lower, and it is 1000,
- the value of hardness is high.

The last parameter is problematic because there are some materials for which this is not measured. This is the case for ADI_42, which matches the other criteria. However, having other examples, the classifier predicting hardness may be trained and applied to this case. Therefore, the expert system recommends ADI_42 with 1.0 confidence.

The proof was obtained by triple application of the Modus Ponens (MP) rule and double application of the object specialization (SPECo) rule. Most of the inference steps were similar to the first scenario. At the beginning, the MP rule was applied to Implication no. 1. Its premises 1, 3, 6, and 9 are user responses. The proof is presented graphically in Figure 3.

Exactly as in the first scenario, Premises 2 and 4 were inferred using the SPECo object specialization rule, because ADI_42 is a typical ADI in terms of both application and cost. Premises 5 and 8 express the idea that the cost inferred (variable COST_CALCULATED) or the tensile strength given by the user (Variable C) fit the

demanded range. Premise 7 was unified with the knowledge base. The last premise related to hardness was missing, and the system was not able to infer it. Therefore, a complex knowledge transmutation was applied, and 14 examples described by all of the available attributes were prepared.

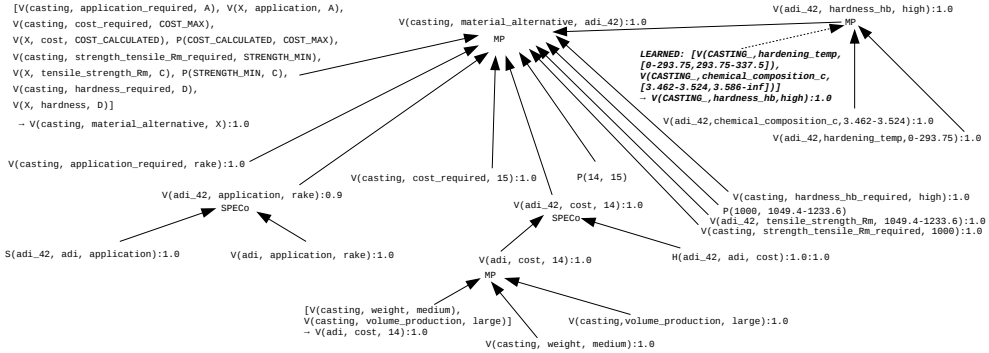

Figure 3. Graphical presentation of proof found in Scenario 2.

One of the rules checked the hardening temperature and carbon content:

$$V(CASTING, hardening\_temp, [0 - 293.75, 293.75 - 337.5]) \\ \wedge\, V(CASTING, chemical\_composition\_c, [3.462 - 3.524, 3.586 - inf]) \\ \rightarrow V(CASTING, hardness\_hb, high) : 1.0. \quad (6)$$

The premises were true for ADI_42 and allowed for the derivation of its hardness (which was supposed to be high), thus recommending ADI_42 to the user.

7.3. Scenario 3

The requirements in this scenario were the same as in the previous one; but, to predict hardness, another implication was selected from the machine learning results.

The proof structure is shown in Figure 4. The inference is similar to the one made in the second scenario. The only difference is replacing the learned implication in the last step with implication:

$$V(CASTING, hardening\_temp, [0 - 293.75, 293.75 - 337.5]) \\ \rightarrow V(CASTING, hardness\_hb, high) : 1.0. \quad (7)$$

It has a shorter form, since the premise related to carbon content is omitted. As a result, ADI_52 was recommended as a casting material with 1.0 confidence. In the previous scenario, this material was not selected because it did not match the condition related to the carbon content.

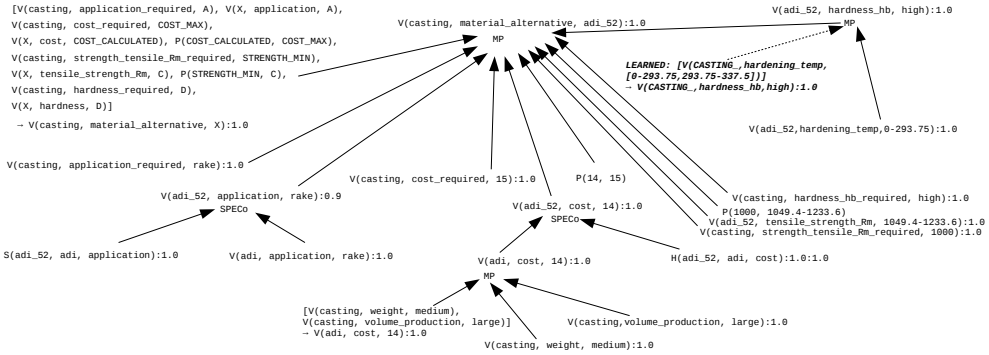

Figure 4. Graphical presentation of proof found in Scenario 3.

7.4. Summary

Domain experts reviewed the results of the presented scenarios. The obtained answers and their certainties were evaluated as correct. The second and third scenarios show us that the use of the MILS model allows for reasoning to be continued even in cases not covered by the knowledge base (KB). In such cases, recommendations depend on the learned implications.

8. Reasoning algorithm time complexity tests

The LPR reasoning process is complex because of a large number of inference rules. The application of machine learning additionally increases the time complexity. Therefore, the performance tests were carried out on the implemented application to check how learning prolongs the computation time.

All tests were performed on a computer with 4 GB of RAM and 4×3.33 GHz processor. Times were measured on the server-side of the application.

The rst test was designed to check how the maximum depth of the proof tree inuences the performance. The inference times for five values of this parameter and two modes of reasoning (i.e., with and without learning) are presented in Figure 5. All times were measured for a scenario that gave meaningful results both with and without machine learning. When complex knowledge transmutations were included, three attributes were allowed as categories: hardness, application, and cost. All proofs with depths less than the allowed maximum one were searched. This corresponds to a pessimistic time complexity, because it is often enough to find the first proof in the expert system mode, which results in a much-shorter calculation time.

The second test was designed to check how the number of possible category attributes influences the performance. The following attributes were added one by one to the category set: hardness_hb, application, cost, tensile strength Rm, hardening_temp, austenization_time, and hardening_time. The inference times for the second scenario and category set size from 1 to 6 are presented in Figure 6.

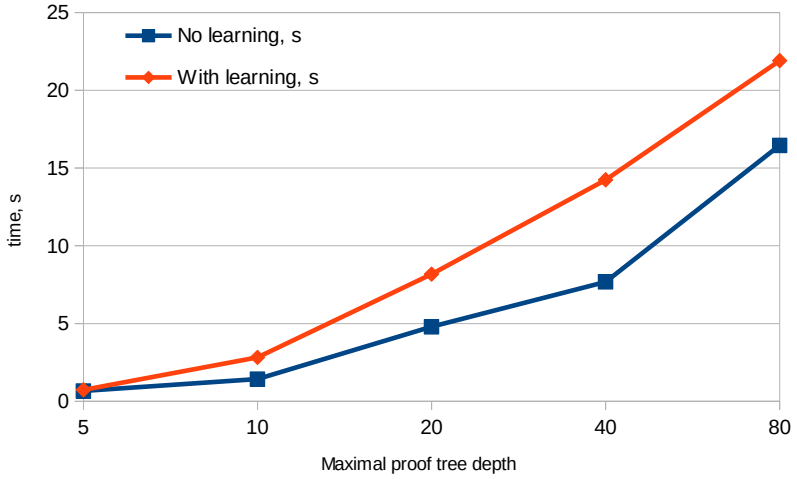

Figure 5. Dependency between inference time and maximum depth of proof tree.

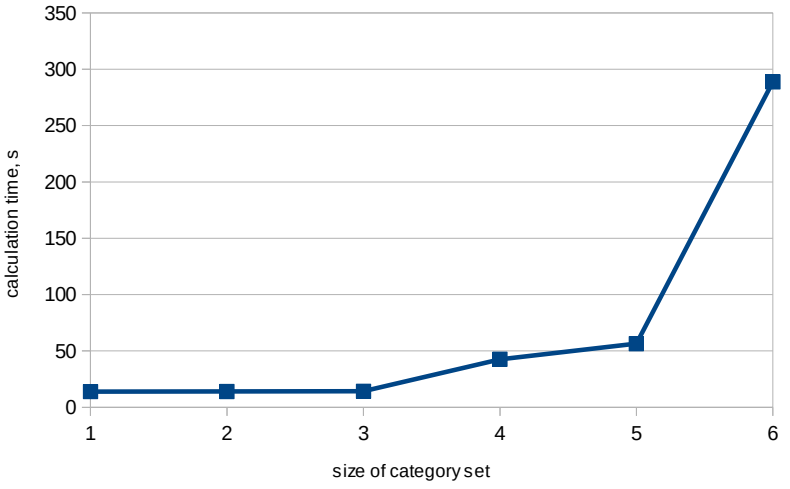

Figure 6. Dependency between inference time and size of category set.

The analysis of the performance test results yielded the following conclusions. The application of machine learning affects the duration of the inference process. For the cases examined, the reasoning was longer by between 10 and almost 100 percent. In spite of this, the results are still acceptable from the users point of view. Another important fact is that, currently, the software is not time-optimized; due to this, the calculation time can be reduced.

An important factor related to the complexity of the machine learning procedure is the number of possible category attributes, which limits the number of complex knowledge transmutations applied. For a xed size of the problem, any increase in this number rapidly escalates the time-consumption. Therefore, the system should be congured in an appropriate way, and only attributes with missing values corresponding to important parameters that are difficult to measure (like hardness) should be selected.

9. Conclusion

The experiments performed on the implemented LIIS system have proven the correct operation of the reasoning algorithm. Its application enables us to design a creative decision support system that, instead of getting stuck in the case when no rule is applicable, automatically creates intrinsically new knowledge to continue the reasoning process.

Our solution can be considered a deductive system that can manage knowledge in a multi-fashion way; i.e., in a similar manner as humans. It combines the search, inference, and machine learning capabilities that are performed in a uniform way, using one reasoning algorithm. As a result, the reasoning process is more creative than in the classical AI models. Depending on the agent experience and context, other knowledge may be discovered from the stored statements.

Further works will concern the enrichment of software capabilities by extending the range of implemented knowledge transmutations. For example, it is planned to add a clustering algorithm that will be used to derive similarity formulas. Testing the system in other domains is also considered. The system should also be optimized to improve its performance.

Acknowledgements

The research reported in this paper was supported by a grant from The National Center for Research and Development (LIDER/028/593/L-4/12/NCBR/2013).

References

[1] Abseher M., Musliu N., Woltran S.: Improving the Efficiency of Dynamic Programming on Tree Decompositions via Machine Learning, *Journal of Artificial Intelligence Research*, vol. 58, pp. 829–858, 2017.

[2] Alkharouf N.W., Michalski R.S.: Multistrategy Task-adaptive Learning Using Dynamically Interlaced Hierarchies. In: Michalski R.S., Wnek J. (eds.), *Proceedings of the Third International Workshop on Multistrategy Learning*, 1996.

[3] Bach K., Deutsch J.O., Hanft A., Manz J., Muller T., Newo R., Reichle M., Schaaf M., Weis K.H., Althor K.D.: Collaborative multi-expert-systems. In: *Proceedings of the International Conference on Artifcial Intelligence*, 2007.

[4] Boehm-Davis D., Dontas K., Michalski R.S.: *Plausible Reasoning: An Outline of Theory and the Validation of its Structural Properties*, North Holland, 1990.
[5] Boehm-Davis D., Dontas K., Michalski R.S.: *A Validation and Exploration of the Collins-Michalski Theory of Plausible Reasoning.* Reports of the Machine Learning and Inference Laboratory. George Mason University, 1990.
[6] Buczak A.L., Guven E.: A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection, *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 1153–1176, 2016.
[7] Collins A., Michalski R.S.: The Logic of Plausible Reasoning: A Core Theory, *Cognitive Science*, vol. 13, pp. 1–49, 1989.
[8] Esterline A.C., Wiriyacoonkasem S.: Adaptive learning expert systems. In: *Proceedings of the IEEE, Southeastcon*, 2000.
[9] Gabbay D.M.: *LDS – Labeled Deductive Systems.* Oxford University Press, 1991.
[10] Hancock J.P., Tran L.P.: An adaptive-learning expert system for maintenance diagnostics. In: *Proceedings of the National Aerospace and Electronics Conference*, 1989.
[11] Hart P., Nilsson N., Raphae J.B., Hart P.: A Formal Basis for the Heuristic Determination of Minimum Cost Path, *Transactions on System Science and Cybernetics*, vol. 4, pp. 100–107, 1968.
[12] Hieb M.R., Michalski R.S.: *A Knowledge Representation System Based on Dynamically Interlaced Hierarchies: Basic Ideas and Examples.* Reports of the Machine Learning and Inference Laboratory. George Mason University, 1993.
[13] Hieb M.R., Michalski R.S.: *Multitype Inference in Multistrategy Task-Adaptive Learning: Dynamic Interlaced Hierarchies.* Reports of the Machine Learning and Inference Laboratory. George Mason University, 1993.
[14] Ho Chung L., Ah Hwee T., Hoon Heng T., Boon Toh L.: Connectionist expert system with adaptive learning capability, *Knowledge and Data Engineering*, vol. 3, pp. 200–207, 1991.
[15] Horzyk A.: Human-Like Knowledge Engineering, Generalization, and Creativity in Artificial Neural Associative Systems. In: Skulimowski A.M.J., Kacprzyk J. (eds.), *Knowledge, Information and Creativity Support Systems: Recent Trends, Advances and Solutions*, *Advances in Intelligent Systems and Computing*, vol. 364, pp. 39–51, Springer, 2015.
[16] Kowalski R.: *Logic for Problem Solving.* Oxford, New York, 2002.
[17] Larson J., Michalski R.S.: *Aqval/1 (aq7) user's guide and program description.* University of Illinois, Urbana, 1975.
[18] Ligeza A.: *Logical Foundations for Rule-Based Systems.* Springer, Berlin–Heidelberg, 2nd edition, New York, 2006.
[19] Michalski R.S.: *Inferential Theory of Learning: Developing Foundations for Multistrategy Learning.* Morgan Kaufmann Publishers, 1994.
[20] Morgan C.G.: Autologic. In: *Logique et Analyse*, vol. 28, pp. 257–282, 1985.
[21] Neapolitan R.E.: *Probabilistic Reasoning in Expert Systems: Theory and Algorithms.* CreateSpace Independent Publishing Platform, USA, 2012.

[22] Ozay M., Esnaola I., Yarman Vural F.T., Kulkarni S.R., Poor H.V.: Machine Learning Methods for Attack Detection in the Smart Grid, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, pp. 1773–1786, 2015.

[23] Quinlan J.R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, USA, 1993.

[24] Riley G.: *Clips – an expert system building tool.* CA, San Jose, 2001.

[25] Roldan Reyes E., Negny S., Cortes Robles G., Le Lann J.M.: Improvement of online adaptation knowledge acquisition and reuse in case-based reasoning: Application to process engineering design, *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 1–16, 2015.

[26] Sniezynski B.: Integration of inference and machine learning as a tool for creative reasoning. Modeling Changing Perspectives – Reconceptualizing Sensorimotor Experiences. Physica-Verlag, Springer.

[27] Sniezynski B.: Probabilistic Label Algebra for the Logic of Plausible Reasoning. In: Klopotek M., Wierzchon S., Michalewicz M. (eds.), *Intelligent Information Systems*, Advances in Soft Computing, Physica-Verlag, Springer, 2002.

[28] Sniezynski B.: Proof Searching Algorithm for the Logic of Plausible Reasoning. In: Klopotek M. (ed.), *Intelligent Information Processing and Web Mining*, Advances in Soft Computing, pp. 393–398. Springer, 2003.

[29] Sniezynski B.: Recommendation System Using Multistrategy Inference and Learning. In: Szczepaniak P.S., Kacprzyk J., Niewiadomski A. (eds.), *Advances in Web Intelligence. AWIC 2005*, Lecture Notes in Computer Science, vol. 3528, Springer, Berlin–Heidelberg, 2005.

[30] Xindong W., Huanhuan C., Gongqing W., Jun L., Qinghua Z., Xiaofeng H., Aoying Z., Zhong-Qiu Z., Bifang W., Ming G., Yang L., Qiping Z., Shichao Z., Ruqian L., Nanning Z.: Knowledge Engineering with Big Data, *IEEE Intelligent Systems*, vol. 30, pp. 46–55, 2015.

[31] Zadeh L.A.: Fuzzy sets, *Information and Control*, vol. 8, pp. 338–353, 1965.

Affiliations

Dorota Wilk-Kołodziejczyk
AGH University of Science and Technology, Faculty of Metals Engineering and Industrial Computer Science, Poland, dorota.wilk@agh.edu.pl, Foundry Research Institute, Krakow, Poland

Received: 9.02.2017
Revised: 16.06.2017
Accepted: 16.06.2017