Proposal of a Graph-Oriented Approach to Verification of XTT2 Rule Bases*

Agata Ligęza, Grzegorz J. Nalepa, Antoni Ligęza

Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
jaga@student.agh.edu.pl, gjn@agh.edu.pl, ligeza@agh.edu.pl

Abstract. The main focus of this paper is on the discussion of formal rule verification using graph-based approach. XTT2 is a custom rule representation method, that introduces a structured rule base composed of extended decision tables linked in a tree-like structure. Considering the complex nature of the XTT2 structure, only the local, table level verification has been considered so far. However, graph-oriented verification is a powerful solution to the analysis of rule-based systems. It can be applied to provide global verification of the XTT2 knowledge bases. The principal idea consists in representing XTT2 rules as a directed hypergraph. All of rule formulas are transformed into vertices and appropriate hyperarcs are determined. This restructuring of the XTT2 knowledge base allows to provide verification using graph algorithms. Preliminary evaluation of this approach shows that the graph-oriented verification is a promising solution to provide a formal analysis of XTT2 rules.

1 Introduction

Rule-based systems (RBS) [1] are an important class of intelligent systems [2]. Their formal description allows for a formal analysis of important system properties. Therefore, it is possible to ensure their quality and safety at the early design stages.

The main focus of the paper is the formal verification of rules [3,4]. Formal rule properties have to be considered w.r.t. to a given knowledge formalization format. Therefore, the rule formalization for the XTT representation is given [5,6]. The representation introduces a structured rule base composed of extended decision tables linked in a tree-like structure. The rule formalization is given using the ALSV(FD) logic [1,7]. Considering the complex nature of the XTT knowledge base structure, so far only local, table level verification has been provided.

The approach proposed in this paper uses graph-based representation. Graph-oriented verification is a powerful solution to the analysis of rule-based systems. It can be applied to provide global verification of the XTT2 knowledge bases.

* The paper is carried out within the AGH UST Project No. 10.10.120.105.

The principal idea consists in representing XTT2 rules as a directed hypergraph. All of rule formulas are transformed into vertices and appropriate hyperarcs are determined. This restructuring of the XTT2 knowledge base allows to provide a verification using graph algorithms. Preliminary evaluation of this approach shows that the graph-oriented verification is a promising solution to provide a formal analysis of XTT2 rules. In the paper a practical example is provided.

2 XTT2 Rule Language Formalization

The formalization for XTT^2 representation is based on ALSV(FD) logic [1,7,6]. The ALSV(FD) provides a much higher expressive power than the propositional calculus, while providing tractable inference. Therefore, a format of rule is more complex. In a general case, rule expressed by attributive logic is represented as (1)

$$\begin{aligned}\mathrm{rule}(i) : \psi\ \wedge\ & \\ & A_1 \in t_1 \wedge A_2 \in t_2 \wedge \cdots \wedge A_n \in t_n \\ & \longrightarrow \\ & retract(B_1 = b_1, B_2 = b_2, \ldots, B_b = b_b) \\ & assert(C_1 = c_1, C_2 = c_2, \ldots, C_c = c_c) \\ & H_1 = h_1, H_2 = h_2, \ldots, H_h = h_h \\ & next(j),\ \ else(k)\end{aligned} \tag{1}$$

In (1) a formula ψ describes context. $A_1 \in t_1 \wedge A_2 \in t_2 \wedge \cdots \wedge A_n \in t_n$ is a precondition formula. C_j $(j = 1, \ldots, c)$ and B_i $(i = 1, \ldots, b)$ correspond to facts to assert and to retract from the knowledge base. Conclusions (decisions or actions) are represented by $H_1 = h_1, H_2 = h_2, \ldots, H_h = h_h$. There is also a control statement introducing the next or alternative rule.

In the case of XTT^2 representation, the logical rule format is as follows:

$$r\colon (A_1 \propto_1 V_1) \wedge (A_2 \propto_2 V_2) \wedge \cdots \wedge (A_n \propto_n V_n) \longrightarrow RHS, \tag{2}$$

where $\propto_i \in \{=, \neq, \in, \notin\}$ for simple attributes (taking single values) and $\propto_i \in \{=, \neq, \subseteq, \supseteq, \sim, \not\sim\}$ for general attributes (taking set values). The form of RHS is as in (1). For more details on the XTT2 rule formalization using ALSV(FD) see [7,6].

3 Basic Formal Analysis of XTT Rules

The quality of a rule-based system is dependent on the quality of a knowledge base. What is more important, anomalies in the set of rules could result in serious faults in system's responses. Therefore the analysis of knowledge base is an important step during development of a rule-based system.

The issues of verification and validation were discussed by many authors. Differences in their approaches to V&V start at the definition level. In this paper verification and validation processes are defined as follows:

Verification is a process in early design phase, aimed at checking if the system meets its constraints and requirements ([8,9,10,11,12]).

Testing is a process aimed at analyzing the system work, by comparing system responses to known responses for special input data ([8]).

Validation is a case of testing, aimed at checking if the system meets user's requirements ([8]).

A summary of analysis techniques and tools is presented in [13].

The classification of potential errors and deformation of knowledge base was also widely discussed, with some practical taxonomies of anomalies given (see [14,1]). In this paper, from the formal point of view, rule anomalies are be divided into three main categories: 1) *incompleteness*, 2) *indeterminism*, and 3) *overdeveloped set of rules*.

Let the knowledge base be described by rules:

$$\begin{array}{l} r_1\colon\ \Psi_1 \rightarrow h_1 \\ r_2\colon\ \Psi_2 \rightarrow h_2 \\ \qquad\vdots \\ r_n\colon\ \Psi_n \rightarrow h_n \end{array} \tag{3}$$

Completeness ensures that for any input state the system reacts and produces some response (conclusion, decision or action) ([15]). In other words, the system with the set of rules (3) is *logically complete* if a disjunction of preconditions is a tautology: $\models \Psi_1 \ \vee\ \Psi_2 \ \vee\ \ldots\ \vee\ \Psi_n$ The knowledge base is incomplete, when *unreachable*, or *dead-end* rules exist in a rule set, or some rules are *missing*.

Determinism guarantees that the system always produces the same reaction for the same input data. In other words for any input state the system finds a unique solution ([15]). From the formal point of view the set (3) is indeterministic "if there exists a state described by formula ψ, such that simultaneously $\phi \models \Psi_1$ and $\phi \models \Psi_2$ and $h_1 \neq h_2$" ([16]).
The system is indeterministic, if there are *contradictory rules* in knowledge base. *Inconsistency* also is a cause of indeterminism.

Minimal number of rules indicates a set of rules without *redundant*, *subsumed* rules. What is more, the set of rules should produce the same reactions as a overdeveloped set.

All of above features – completeness, determinism, minimal number of rules – should be provided to assure reliability, safety and efficiency of the rule-base system ([15]). The XTT^2 representation introduces a structure of knowledge base by identifying *rule contexts*. Implicitly the context is identified with an extended decision table. Isolation of contexts allows to provide *local analysis* – contexts can be verified separately. In practice, the analysis of the XTT knowledge base is provided by HalVA Verification Framework [17]. The main purpose of the framework is the local verification. HalVA in implemented in Prolog. The verification framework was developed as a plug-in of the HeaRT inference engine [18]. The

HalVA provides the verification of completeness, contradiction and subsumption. What is more, the number of rules can be reduced. HalVA verification is focused on a local level, the schema of the XTT table is considered.

All of verification procedures are based on inference rules for ALSV(FD) introduced in [19]. For state described by ϕ a precondition $(A_i \propto_i V_i)$ (where $\propto_i \in \{=, \neq, \in, \notin\}$ for the simple attribute and $\propto_i \in \{=, \neq, \subseteq, \supseteq, \sim, \not\sim\}$ for the general attribute, $i = 1, \ldots, n$) is satisfied, if simultaneously:

- V_i is a value from the domain of A_i,
- ϕ_{A_i} is a value from the domain of A_i, where ϕ_{A_i} is a value of attribute A_i in formula ϕ,
- a formula $(A_i = \phi_{A_i})$ is a logical consequence of a formula $(A_i \propto_i V_i)$.

The basic idea in the verification of *completeness* consists in checking all input states. Domains of attributes are considered. The Cartesian product of domains determine all states for the context (table). For every tuple corresponding to an input state, the algorithm checks if preconditions of any rule are satisfied. If there is no rule to execute, the considered state is reported as uncover. Based on all uncovered states, a proposal of a new rule is given. The analysis ends, when all states are checked. Since domains of attributes are finite, the verification procedure terminates after a finite number of discrete steps.

Verification of *contradiction* is based on a pairwise comparison of rules. Two rules, executable in the same time (for the same state), are taken into consideration. The comparison concerns the right-hand side of rules. If conclusions are inconsistent, the conflict is reported. The verification procedure stops when all possible comparisons are done.

The pairwise comparison of rules is also used to verify *subsumption*. However, the analysis concerns both sides of rules. One rule is subsumed by another, if its preconditions are more specific, but simultaneously conclusions are more general. The verification procedure finds two rules in the context (table), executable for the same state. Then checks, whether there is relation between conclusions. The algorithm provides all comparisons. This strategy allows to detect identical rules. In this case, a rule is reported as subsuming another and the other subsuming the first one.

HalVA allows to reduce an overdeveloped set of rules. The reduction can be done by using the dual resolution ([16]). If rules produce the same conclusions and in the precondition part exists at least one the same formula, the remaining formulas are joined into one. All possible reductions are reported. What is more, proposals of new rules are introduced.

Unfortunately, solutions provided by HalVA are limited. The serious issue is computational complexity of provided algorithms. Verification of completeness could cause a combinatorial explosion, if domains of attributes are outsized. The other technique – the pairwise comparison of rules – is also dependent on a size of the considered case. Generally, to verify a set of n rules, $\binom{n}{2}$ comparisons need to be done. What is most important, all of HalVA verification features are focused on a local analysis. The limitation of the scope indicates a verification of some context (implicitly a table) only. Therefore, HalVA procedures can point out anomalies in some specific area. Unfortunately, the problem of the global quality of the knowledge base remains unsolved. This is where the new approach proves to be useful.

4 Graph-oriented Verification Solution Proposal

To verify rule-based systems in global scope the graph-oriented approach is introduced. The main concept consists in representing XTT2 rules as a directed hypergraph. Necessary basic definitions are introduced bellow.

"Let $X = \{x_1, x_2, \ldots, x_n\}$ be a finite set, and let $\mathcal{E} = (E_i | i \in I)$ be a family of subsets of X. The family $\mathcal{E}$ is said to be a *hypergraph on X* if

1. $E_i \neq \emptyset \ (i \in I)$
2. $\bigcup_{i \in I} E_i = X$.

The couple $H = (X, \mathcal{E})$ is called a *hypergraph.* $|X| = n$ is called the *order* of this hypergraph. The elements $x_1, x_2, \ldots, x_n$ are called the *vertices* and the sets $E_1, E_2, \ldots, E_m$ are called the *edges* ([20])."

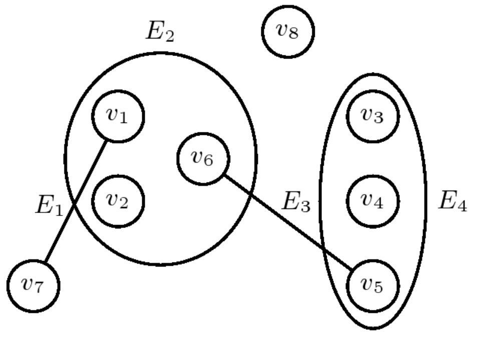

Fig. 1. A hypergraph: $X = \{v_1, \ldots, v_8\}$ – vertices, $\mathcal{E} = \{E_1, \ldots, E_4\}$ – edges, where: $E_1 = \{v_1, v_7\}$, $E_2 = \{v_1, v_2, v_6\}$, $E_3 = \{v_5, v_6\}$ and $E_4 = \{v_3, v_4, v_5\}$.

A hypergraph is presented in Fig. 1. For *directed* edges (hyperarcs) *initial* and *terminal* endpoints can be pointed out (vide Fig. 2).

Let a hypergraph $G = (X, \mathcal{E})$ be considered (vide Fig. 3). "The *outer demi-degree* $d_G^+(x)$ of a vertex x is defined as the number of arcs having x as their initial endpoint ([20])." On the other hand, "the *inner demi-degree* $d_G^-(x)$ of a vertex x is defined as the number of arcs having x as their terminal endpoint ([20])." A vertex x is said to be a *source*, if $d_G^-(x) = 0$. A vertex x is said to be a *sink*, if $d_G^+(x) = 0$.

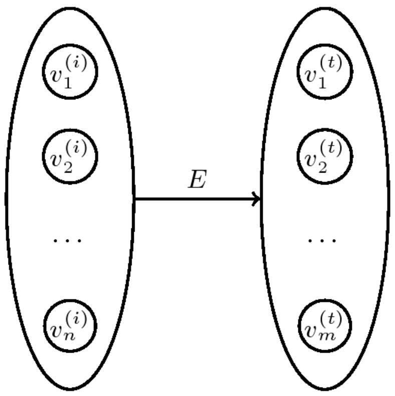

Fig. 2. A hyperarc $E = \{v_1^{(i)}, \ldots, v_n^{(i)}, v_1^{(t)} \ldots, v_m^{(t)}\}$. Vertices: $v_1^{(i)}, \ldots, v_n^{(i)}$ are initial endpoints and $v_1^{(t)}, \ldots, v_m^{(t)}$ are terminal endpoints of the hyperarcs

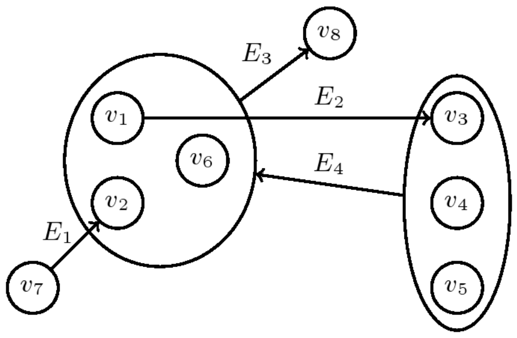

Fig. 3. A directed hypergraph. Degrees of v_1: $d^+(v_1) = 2$, $d^-(v_1) = 1$. Vertices v_4, v_5, v_7 are sources. A vertex v_8 is a sink

For a directed hypergraph $H = (X, \mathcal{E})$ the *incidence matrix* is a matrix $((a_{ij}))_{m \times n}$ "with m rows that represent the edges of H and n columns that represent the vertices of H, such that" ([20]):

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \in E_i \text{ and } x_j \text{ is an inital endpoint of } E_i \\ 1 & \text{if } x_j \in E_i \text{ and } x_j \text{ is an terminal endpoint of } E_i \\ 0 & \text{if } x_j \notin E_i \end{cases}$$

Incidence matrix for the hypergraph in the Fig. 3 is introduced in Table 1.

To provide verification using the graph theory an appropriate transformation of XTT rules needs to be done. The first step is to determine vertices. If vertices correspond to all possible values from domains of attributes, it causes a combinatorial explosion for outsized domains. Therefore, not single values, but whole

Table 1. The incidence matrix for the hypergraph in the Fig. 3.

	E_1	E_2	E_3	E_4
v_1	0	-1	-1	1
v_2	1	0	-1	1
v_3	0	1	0	-1
v_4	0	0	0	-1
v_5	0	0	0	-1
v_6	0	0	-1	1
v_7	-1	0	0	0
v_8	0	0	1	0

formulas are transformed into vertices. However, using formulas to describe vertices results in an additional assumption. The graph-oriented approach can be provided, if the table-level verification of completeness is done.

There are three classes of vertices in the graph representation of rules:

– *sources*: the source corresponds to a formula $(A_i \propto_i V_i)$, where the attribute A_i is used only in a precondition part of rules.
– *sinks*: the sink corresponds to conclusion part of rules.
– *others*: all of others vertices, which are neither source nor the sinks, are related to formulas $(A_i \propto_i V_i)$, where attribute A_i appears in both a precondition and a conclusion part of rules.

The assumption of local verification of completeness ensures the absence of isolated vertices. Therefore, for the source x: $d^+_G(x) > 0$, for the sink x: $d^-_G(x) > 0$, and for *others* x: $d^+(x) > 0$ and $d^-(x) > 0$. In this representation, a rule from the knowledge base is transformed into a hyperarc. The rule also determines the direction of the hyperarc. Therefore, each rule has a unique representation. Initial endpoints of the hyperarc are related to formulas in a precondition part of the rule. The hyperarc is terminated in several ways. Firstly, if a conclusion in the rule is related to a sink, the sink becomes a terminal endpoint of the hyperarc. Secondly, if an attribute in a conclusion appears in a clause in a precondition part of other rule and if the clause is more general than the conclusion, the vertex related to the clause terminates the hyperarc. Finally, if control statements (*next*, *else*) are introduced in the rule, appropriate vertices (clauses) become terminators of the hyperarc.

Table 2. An example of a XTT2 system

r_1	A in $[a_1, a_2]$	B set b_1	C set c_1
r_2	A eq a_3	B set b_2	C set c_2

r_3	C eq c_1	D set d_1

The hypergraph for the system is presented in Fig. 4. Let the XTT2 system be described by rules r_1, r_2 and r_3 (vide Table 2). The attribute A appears only in a precondition part. On the other hand, attributes B and D appear in a decision part. Therefore, a hypergraph for the system has vertices:

v_1: $(A \text{ in } [a_1, a_2])$	v_3: $(B \text{ set } b_1)$	v_5: $(C \text{ eq } c_1)$	v_7: $(D \text{ eq } d_1)$
v_2: $(A \text{ eq } a_3)$	v_4: $(B \text{ set } b_2)$	v_6: $(C \text{ set } c_2)$	

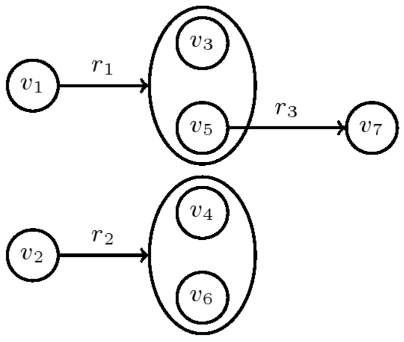

Fig. 4. A directed hypergraph for the subsystem

The introduced representation of rules allows to define anomalies of the knowledge base as follows ([21]):

Inconsistency – exists if there exists a path from vertex X to its exclusive vertex $\neg X$ ([21]).
Contradiction – exists if two paths from vertex X to vertices Y and $\neg Y$ exist in the graph.
Redundancy – exists if at least two different paths from vertex X to Y exist.
Circularity – exists if there is a cycle in the graph.
Unreachability – exists if a vertex which is not the beginning of a path to any output node and is not the end of a path from any input node can be found in the graph.

Locally verified inconsistency, contradiction and redundancy assure correctness of rules in the table scope only. The new representation of rules allows to examine paths in the graph. Deformed ones are reported. However, the vital question is the global verification of completeness. Local analysis of verification is helpful, but it is not limitative in the global scope. Fortunately, graph-oriented approach allows to detect *unreachable formula* or *dead-end formula* – main sources of incompleteness.

As it was mentioned earlier, if the vertex $v \in V$ is neither the source nor the sink, its outer and inner demi-degree are greater than zero. If $d^+(v) = 0$ and

$d^-(v) > 0$, the vertex is a dead-end formula. On the other hand, if $d^+(v) > 0$ and $d^-(v) = 0$, the vertex is a unreachable formula.

The graph-oriented verification is a powerful solution to the analysis of rule-based systems. It can be applied to provide a global verification of the XTT2 knowledge bases ([5,22,6,23]). The most important feature of this approach is its ability to verify completeness in a global scope.

5 Verification Example

Consider the thermostat control system ([16]). The general schema of the system is presented in Fig. 5. Tables *dt* and *th* are considered. The subsystem will be transformed into a hypergraph (vide Fig. 6).

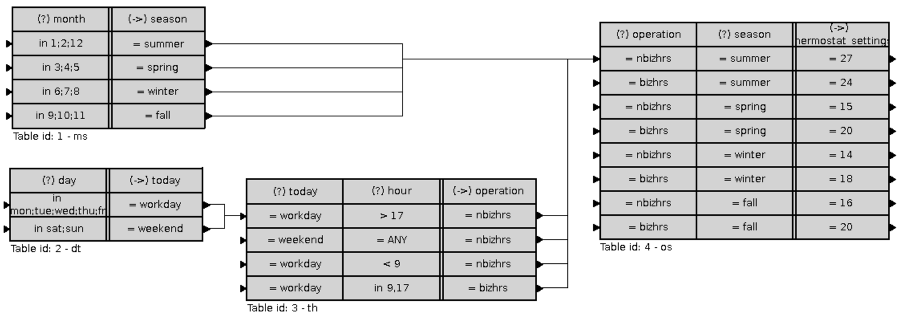

Fig. 5. The general schema of the thermostat control system

Table 3. The incidence matrix for the thermostat control system

vertex	formula	dt/1	dt/2	th/1	th/2	th/3	th/4
1	day in [mon;tue;wed;thu;fri]	-1	0	0	0	0	0
2	day in [sat;sun]	0	-1	0	0	0	0
3	today eq workday	1	0 (**1**)	-1	0	-1	-1
4	today eq weekend	0	1 (**0**)	0	-1	0	0
5	hour gt 17	0	0	-1	0	0	0
6	hour eq any	0	0	0	-1	0	0
7	hour lt 9	0	0	0	0	-1	0
8	hour in [9 to 17]	0	0	0	0	0	-1
9	op eq nbizhrs	0	0	1	1	1	0
10	op eq bizhrs	0	0	0	0	0	1

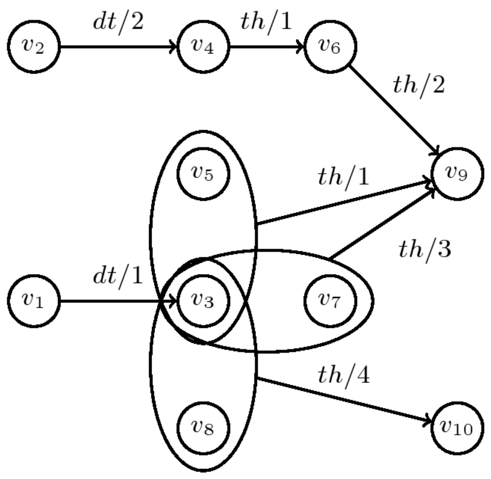

Fig. 6. A directed hypergraph for the subsystem

The hypergraph related to the subsystem is presented as an incidence matrix (vide Table 3). Let the deformation during the knowledge acquisition phase be introduced. The altered knowledge base could be described by hypergraph where vertices 3 and 4 are modified as shown in the parentheses. Locally the knowledge base is still valid: it is complete and there is no contradiction or redundancy. However, consider the global scope: vertices 1, 2, 5, 6, 7 and 8 are sources; vertices 9 and 10 are sinks. Demi-degrees (inner and outer) of other vertices – 3 and 4 – should be greater than zero. Unfortunately, the vertex 4 is deformed. Its inner demi-degree equals zero. There is no path from any source to this vertex. It is an *unreachable formula*. Therefore, the knowledge base is globally incomplete.

6 Concluding Remarks and Future Work

In this paper a graph-based representation for XTT2 rules is proposed. It is aimed at the graph-oriented verification, which is a powerful solution to the analysis of rule-based systems. It can be applied to provide global verification of the XTT2 knowledge bases. In the paper a practical example is provided, and a preliminary evaluation of the approach is given.

In the global verification of XTT2, formulas in precondition and decision parts are transformed into vertices. Adequate hyperarcs are determined. The analysis of hypergraph structure allows to detect unreachable and dead-end formulas. This allows for a more efficient analysis of completeness of the rule base.

The research presented in the paper is work in progress. Detection of redundancy or contradiction needs to examine paths in the hypergraph. This is a more difficult problem than the analysis of structure of the hypergraph. The correlation between vertices on a path needs to be checked. Therefore, this method should be improved. Another important question is reconstructing the system from the hypergraph. At the moment the transformation to hypergraph is not entirely reversible. Yet another issue is the global inference support based on the hypergraph structure. Apparently, in this approach the inference can be optimized on both the table level and global level.

References

1. Ligęza, A.: Logical Foundations for Rule-Based Systems. Springer-Verlag, Berlin, Heidelberg (2006)
2. Hopgood, A.A.: Intelligent Systems for Engineers and Scientists. 2nd edn. CRC Press, Boca Raton London New York Washington, D.C. (2001)
3. Liebowitz, J., ed.: The Handbook of Applied Expert Systems. CRC Press, Boca Raton (1998)
4. Giarratano, J., Riley, G.: Expert Systems. Principles and Programming. Fourth edition edn. Thomson Course Technology, Boston, MA, United States (2005) ISBN 0-534-38447-1.
5. Nalepa, G.J., Ligęza, A.: A visual edition tool for design and verification of knowledge in rule-based systems. Systems Science **31**(3) (2005) 103–109
6. Nalepa, G.J., Ligęza, A.: Hekate methodology, hybrid engineering of intelligent systems. International Journal of Applied Mathematics and Computer Science (2009) accepted for publication.
7. Nalepa, G.J., Ligęza, A.: Xtt+ rule design using the alsv(fd). In Giurca, A., Analyti, A., Wagner, G., eds.: ECAI 2008: 18th European Conference on Artificial Intelligence: 2nd East European Workshop on Rule-based applications, RuleApps2008: Patras, 22 July 2008, Patras, University of Patras (2008) 11–15
8. Tepandi, J.: Verification, testing, and validation of rule-based expert systems. Proceedings of the 11-th IFAC World Congress (1990) 162–167
9. Andert, E.P.: Integrated Knowledge-Based System Design and Validation for Solving Problems in Uncertain Environments. International Journal of Man-Machine Studies **36**(2) (1992) 357–373
10. Nazareth, D.L.: Issues in the Verification of Knowledge in Rule-Based Systems. International Journal of Man-Machine Studies **30**(3) (1989) 255–271
11. Nguyen, T.A., Perkins, W.A., Laffey, T.J., Pecora, D.: Checking an Expert Systems Knowledge Base for Consistency and Completeness. In: IJCAI. (1985) 375–378
12. Suwa, M., Scott, C.A., Shortliffe, E.H.: Completeness and consistency in rule-based expert system. In: Rule-Based Expert Systems. Addison-Wesley Publishing Company (1985) 159–170
13. Wentworth, J.A., Knaus, R., Aougab, H.: Verification, Validation and Evaluation of Expert Systems. World Wide Web electronic publication: http://www.tfhrc.gov/advanc/vve/cover.htm
14. Vermesan, A.I., Coenen, F., eds.: Validation and Verification of Knowledge Based Systems. Theory, Tools and Practice. Kluwer Academic Publisher, Boston (1999)
15. Ligęza, A.: Logical Support for Design of Rule-Based Systems. Reliability and Quality Issues. ECAI'96 Workshop on Validation, Verification and Refinement of Knowledge-Based Systems (1996)

16. Ligęza, A.: Logical Foundations for Rule-Based Systems. Uczelniane wydawnictwa Naukowo-Dydaktyczne AGH w Krakowie (2005)
17. Ligęza, A., Nalepa, G.J.: Proposal of a formal verification framework for the xtt2 rule bases. In: CMS'09: Computer Methods and Systems 26–7 November 2009, Kraków, Poland, AGH University of Science and Technology, Cracow, Oprogramowanie Naukowo-Techniczne (2009)
18. Nalepa, G.J., Ligęza, A., Kaczor, K., Furmańska, W.T.: Hekate rule runtime and design framework. In Adrian Giurca, Grzegorz J. Nalepa, G.W., ed.: Proceedings of the 3rd East European Workshop on Rule-Based Applications (RuleApps 2009) Cottbus, Germany, September 21, 2009, Cottbus, Germany (2009) 21–30
19. Nalepa, G.J., Ligęza, A.: HeKatE Methodology, Hybrid Engineering of Intelligent Systems. Int. J. Appl. Math. Comput. Sci. **18**(3) (2009) 1–15
20. Berge, C.: Graphs and Hypergraphs. Elsevier Science Ltd (1985)
21. Arman, N., Richards, D., Rine, D.: Structural and Syntactic Fault Correction Algorithms in Rule-Based Systems. International Journal od Computing and Information Sciences **2**(1) (2004)
22. Nalepa, G.J., Ligęza, A.: XTT+ Rule Design Using the ALSV(FD). In: RuleApps. (2008)
23. Nalepa, G.J., Ligęza, A.: On ALSV Rules Formulation and Inference. In: FLAIRS Conference. (2009)