AVANT, Vol. VI, No. 2/2016
ISSN: 2082-6710 avant.edu.pl/en
DOI: 10.26913/70202016.0112.0006

Diversity of Rule-based Approaches: Classic Systems and Recent Applications

Grzegorz J. Nalepa
Jagiellonian University
AGH University of Science and Technology
grzegorz.j.nalepa@uj.edu.pl

Abstract

Rules are a common symbolic model of knowledge. Rule-based systems share roots in cognitive science and artificial intelligence. In the former, they are mostly used in cognitive architectures; in the latter, they are developed in several domains including knowledge engineering and machine learning. This paper aims to give an overview of these issues with the focus on the current research perspective of artificial intelligence. Moreover, in this setting we discuss our results in the design of rule-based systems and their applications in context-aware and business intelligence systems.

Keywords: rules; knowledge engineering; learning; context-aware systems; cognitive-architectures

Representing Knowledge with Rules

Rules are considered one of the most natural ways to represent human knowledge. They can be treated as a general form of conditional statement; however, in specific cases they can also be used to express factual knowledge. More importantly, rule interpretation and processing can model several types of human reasoning. These include deductive reasoning related to forward chaining (where conclusions are made upon facts), abductive reasoning corresponding to backwards chaining (where premises satisfying a given goal are identified), as well as inductive reasoning modeled by rule induction (where rules are discovered based on facts). In a way, the original works on rule-based representations can be thought of as belonging to the core research shared by both cognitive science and artificial intelligence (AI) when they began.

In this paper, we briefly discuss these common roots that are related to cognitive architectures in cognitive science and production rule systems in artificial intelligence. While the former grew as increasingly complex and layered systems based on a largely unchanged rule-processing engine, the latter evolved into flexible solutions that are today mostly embedded and integrated in other AI systems.

In AI, rule-based systems (RBS) are built in the knowledge engineering (KE) paradigm. When it is hard to acquire and encode expert knowledge, rule bases can be also built automatically using dedicated machine learning algorithms. These different approaches result in a variety of specific rule representations, languages and systems. Therefore, we will outline today's landscape of rule-related systems and their applications, as well as the challenges they face.

Rules in Cognitive Architectures

The origins of computer programs based on rules that were supposed to model human intelligence are related to the work of Newell and Simon on the early Logic Theorist system and the later General Problem Solver (GPS). The GPS used Mean-Ends Analysis and a general rule-based representation for problem solving (Newell 1972). The solutions proposed in GPS gave the motivation for the development of two important cognitive architectures: SOAR by Newell (1990) and Laird (2012), and ACT-R by Anderson (1983, 1993). They both adopted the Problem-Space Computational Model introduced by Newell (1990), which considers intelligent agents that behave on the basis of knowledge they possess with actions based on a reasoning process, which in turn is a series of decisions based on logical inferences. The inference process is modeled with the use of rules. Cognitive architectures also aim to model human memory and learning. In such a system, working memory or a fact base (corresponding to short-term memory) is used, along with the knowledge (rule) base (corresponding to long-term memory). Using this approach, reasoning and decision-making can be achieved. Such systems are also capable of planning using backwards reasoning. Furthermore, rule learning can be achieved, e.g. using a mechanism called chunking in SOAR.

Cognitive architectures play a very important role in modeling important aspects of human intelligence by cognitive simulation. However, from the perspective of AI they have certain limitations. Firstly, they adopted the original rule-based nucleus of production systems (described next). Therefore, the extensions of reasoning algorithms, knowledge base management and language extensions were not adopted. Furthermore, they are based on the strong assumption that rules are the main (and only) knowledge representation used in intelligent systems. Moreover, rules are not just

a language for expressing knowledge, but also the meta-language responsible for transforming rule-based knowledge. This simplicity can be an important limitation. Current AI approaches are much more flexible, open and heterogeneous and combine various methods for knowledge representation, processing and learning. Some of these will be described next.

Production Rule Systems

AI research has been driven mainly by construction of practical systems and their applications. One of the first AI systems based on rules was Ops5 (Brownston 1985). Essentially, it shared the basic assumptions of the rule-based core of cognitive architectures: iIt had a rule base, a rule interpreter and working memory. It also provided a simple rule language. Rules stored in the rule base were analyzed by the interpreter. When given rule conditions were met, a rule was fired. Since a rule could produce a new fact that would modify the contents of the working memory, these systems were called production rule systems, or sometimes production systems for short. They also shared with cognitive architectures the computational limitations of the rule interpreter.

The basic idea of production rule interpretation is to scan a rule, and check if the condition is met. This happens if the formula in the condition matches a fact stored in the working memory of the system. This is usually called a pattern-matching step. However, in practice, rule conditions can be composed of many formulas called conditional elements. Moreover, there can be tens of thousands of rules and millions of facts in the working memory. This makes the naïve pattern-matching algorithm unfeasible due to the combinatorial explosion.

An important step in the development of the production systems was the development of the Rete algorithm in the 70s (Forgy 1982). Instead of repeating the pattern-matching steps on the contents of the rule base, the algorithm pre-processes rules that try to identify equivalent conditional expressions. On this basis, it builds a two-layered network (thus the name Rete) corresponding to the rule base. The nodes of the network correspond to the repeated formulas. This network is then used in the pattern-matching process. Since it is more compact than the original rule base and its size is much smaller, the inference process can be conducted even in large rule bases. While Rete does not fully overcome the combinatorial explosion, it pushed the limits of the production systems and allowed for its broad adoption in the form of rule-based expert system shells. In fact, several improved versions of Rete-like algorithms have been proposed, including Treat (Miranker 1987) and Gator (Hanson 1993).

Production systems were the starting point for the development of expert systems, e.g. MYCIN (Buchanan 1984). These systems used rules to express the knowledge of human domain experts in order to solve or support decision problems in this domain. As decision models have become more complex, decision support systems have developed beyond the original architecture of expert systems. However, the core of the technology remains and has been successfully used in the form of so-called shells.

An expert system shell is software that implements a rule-based reasoning engine that is able to efficiently process a given rule language. The engine also provides additional interfaces to operating systems and other services. A shell is often stripped of other tools supporting the KE process. A classic example of such a solution is CLIPS (Giarratano 2005), which is a very fast rule engine providing a rich rule language whose syntax builds on the original OPS5 syntax. A more recent re-implementation of CLIPS is Jess (Friedman-Hill 2003), which is implemented in the Java programming language and is thus more portable than CLIPS and much easier to integrate with Java applications. Development of expert systems has also included extensions of the rule language in order to handle imprecise or uncertain knowledge (Liebowitz 2003) and has led to the creation of an important field of AI: knowledge engineering.

Knowledge Engineering with Rules

Currently, knowledge engineering (KE) remains an active field in AI. While it usually has little to do with the development of expert systems, it is constantly developing. Roughly speaking, knowledge engineering is about processes, methods and tools for capturing, representing, using and managing knowledge encoded with symbolic languages, which are most often grounded in logic. For in-depth overviews, refer to (van Harmelen 2007).

From the system development point of view, KE provides a process comprising several phases. These mainly include knowledge acquisition, modeling (design), implementation (e.g. in a specific rule language), and quality assurance. These stages can be supported by dedicated methods and tools. Currently, the KE process is usually aimed not at a complete stand-alone system, but its knowledge-based component, which is usually a knowledge base.

From the language point of view, KE considers many representations: some emphasize the structural aspect of knowledge e.g. Minsky's frames, while others including rules are more aimed at processing. Moreover, some of the representations can have a well-defined logical formulation. In fact, even though rule languages were originally proposed as programming solutions, there has been a lot of research on logical formulations of rule-based systems

(Ligęza 2006). Such approaches allow for the introduction of purely logical means of formal verification of important properties of a system as a whole, or at least the rule base (Vermesan 1999). Verification of properties such as completeness or determinism is important for assuring the safe operation of a system. This is especially important in control applications. In such cases, completeness means that the system makes a decision in all possible situations. A deterministic system will always make the same decision given the same set of premises (input facts). Representations are used to capture and encode knowledge that can be used in a reasoning process. With formalized representations, this process corresponds to logical inference. Thus, the core of this area is often referred to as KRR (Knowledge Representation & Reasoning) (Brachman 2004).

In (Nalepa, Ligęza 2010) we discussed a formalized notation for rule-based systems called ALSV(FD) (Attributive Logics with Set Values over Finite Domains). This is based on the concept of attributes describing the features of the system. Knowledge representations based on attributes are intuitive and follow simple technical interpretations in which the behavior of the physical systems is formalized by providing the values of system variables. This kind of logic is omnipresent in various applications, e.g. attributive decision tables and trees (Klosgen 2002). ALSV(FD) assumes the dynamic system perspective, in which the state of the system is described by the current values of attributes. The transitions between states are triggered where decision rules are fired. Furthermore, two types of attributes are identified: simple ones that take only one value at a time, and generalized ones that take multiple values at a time. This makes it an expressive language suitable not only for rule formulation, but also for formalized verification of the system (Nalepa 2011a et al).

Creation of a rule base is often a complex and time-consuming process. From the design perspective, basic rule representation is not easy to handle. Therefore, KE uses other supporting methods that correspond to rules, of which the two most important are decision tables and decision trees. What makes them especially important from practical point of view is the fact that they both can be converted to a set of decision rules. In this way, it is feasible to use them as a visual design method for rule bases.

A decision table can be described as a matrix that represents a rule set. Rules in the set have the same structure, i.e. they must have the same sets of conditional and decision attributes. Rules can be represented as rows or columns of a decision table (depending on the notation). Moreover, table notation often introduces additional operators, e.g. for missing or unimportant values. Tables are a very transparent engineering notation that makes a set of similar rules more transparent and easy to comprehend. Decision tables can be interpreted directly by a dedicated engine, or translated

to a set of rules. A recent and very significant step in the standardization of decision steps is the proposal of Decision Modeling Notation (DMN) by Vanthienen and the decision models community (OMG 2011). The second important notation is decision trees. In general, each node of a tree corresponds to a given system attribute. The node is split and branches created for different values of the attribute. As such, a tree is a transparent representation that allows the decision making process to be visualized. In fact, a number of variants of decision trees are considered in the literature.

While rule-based shells are mostly used in the implementation phase, the practical design of the rule base requires additional methods and tools that often make use of the aforementioned solutions. In (Nalepa et al. 2001b), we introduced a hybrid representation for complex rule bases called XTT2 (eXtended Tabular Trees version 2), which was later incorporated as a core part of the Semantic Knowledge Engineering approach (Nalepa 2011). XTT2 is a formalized rule language for modularized rule bases composed of decision units. These units can be represented visually as extended decision tables. Each row of the table corresponds to a decision rule built with the ALSF(FD) formulas. Rules in a table are processed sequentially. Tables can be linked to represent the sequence of the inference process to form a decision network. When a given rule fires, the inference control can be transferred to another table. Therefore, the interpreter for XTT2 called HeaRT (Nalepa 2010) provides a context switching inference mode. By default, the inference process can be forward driven. However, goal-oriented inference is also possible. In such a case, a target table is provided and the execution plan for the needed tabled is built automatically. Furthermore, the resulting rule base can be verified on a formal level with respect to important formal properties such as completeness or lack of redundancy of knowledge. The XTT2 method found several successful applications that are described in more detail in (Nalepa 2011).

Learning Rule Bases

The knowledge engineering approach assumes that the knowledge base is designed and built by human knowledge engineers. However, in some cases this can be challenging or barely possible. Machine learning (ML) approaches have been developed to deal with such difficulties. In ML, setting-dedicated algorithms build mathematical models of knowledge needed to solve certain tasks, e.g. classification. Some early work in this area was focused on inductive concept learning, i.e. construction of logical expressions (here concepts) that define a set of objects from examples (thus, this is supervised learning). Currently, many models are considered in ML, including logical ones such as rules and trees (Flach 2012). In fact, the work on decision trees was seen by some as a major step in the earlier days of ML. This includes the

formulation of the tree induction algorithm ID3 by Quinlan (1986), which is a divide-and-conquer algorithm that builds decision trees using the most useful attributes, based on the impurity measure that assesses how effective decision trees are in the classification task.

A typical decision rule-learning algorithm is a covering algorithm (in the sense of covering the search example space) that iteratively builds new rules and then removes the examples covered by them. Rule learning was pioneered by Michalski (1975) with the AQ system. Decision rules are mainly used for predictive tasks such as classification; however, descriptive cases are also considered as they also play an important role. In an unsupervised learning scheme, association rules can be discovered. Such rules are not used for decision making, such as classification. Instead, they capture correlations between features in the data set. A well-known algorithm for association rule learning is the Apriori algorithm (Agraval et al. 1993). The most expressive rule languages in ML are considered in the inductive logic programming approach, in which Horn rules (used for example in the Prolog programming language) can be built automatically.

Recently, the concept of incremental learning has gained a lot of attention. A typical supervised ML method (batch learner) requires re-learning of the model with a change in the training set (batch) of examples. However, this can be computationally intensive when the set of learning data is constantly updated, e.g., when it is based on a stream of experimental data. To solve such problems, a new generation of incremental learning algorithms that learn from data streams has been introduced (Gama 2010). They can be divided into 1) batch learners that were adapted to act as incremental learning algorithms, and 2) purely incremental learners designed for the purpose of knowledge discovery from data streams, i.e. Hoeffding trees (Domingos 2000).

In (Nalepa 2013) we considered an adapted batch learner that could model user habits related to the use of applications on a mobile device, thus providing important energy savings. More recently, in our yet unpublished work we demonstrated the use of Hoeffding trees for similar tasks, in which the tree structure was converted into a set of XTT2 rules. In fact, we proposed an original extension of the algorithm, in which uncertain information can also be handled effectively. A recent and interesting area for applying rules both in KE and ML settings is so-called context-aware systems.

Use of Rules in Context-Aware Systems

The exposure of context is related to the impossibility of representing complete world knowledge in a system. At the same time, human intelligence is often about recognition of what knowledge is relevant in a given situation or state. Therefore, context-aware systems need to recognize a given context to

select relevant knowledge in order to take appropriate action. A common definition of context is "any information that can be used to characterize the situation of an entity" (Dey 2000). This information may be related to the location of the user (spatial context), presence or absence of other devices and users, or collaboration with others (social context), time (temporal context), as well as user activity. Contextual data can be provided in pervasive computing environments. Machine learning methods can be used to gain knowledge about patterns of human behavior, including daily habits and routine changes in the environment.

Representing contextual knowledge is non-trivial. Models of such knowledge should be scalable, transparent to the user, and possible to improve using new data. In (Nalepa 2013) we proposed a learning middleware architecture based on XTT2 rules which is able to learn from environmental data. This architecture is aimed at context-aware systems that run on mobile devices, such as smart phones. In this case, the environmental data can be acquired in real time from different sensors. What makes building such systems a challenge is their high dynamics. Sensor readings may be temporarily unavailable or inaccurate, thus introducing uncertainty. This has to be represented in the knowledge model. In (Nalepa, Bobek 2014) we proposed an extension of the XTT2 rule base that allows Certainty Factors to be introduced. By using them, it is possible to represent knowledge of different quality. Moreover, system decisions such as recommendations can be given with a level of certainty that can be improved over time with additional data.

Rule Diversity

The discussion provided so far clearly demonstrates that the current landscape of rule-based systems is rich and diverse in many ways. Firstly, a number of different rule types have to be identified to understand their possible interpretation and use. The RuleML Consortium (http://ruleml.org) identifies some of the most important types of rules. Combined with what has been said about the ML perspective, these can be the following types of rules:

1. production – with forward chaining, i.e. data-driven reasoning, from a logical point of view is related to deduction,
2. reaction – similar to the above, firing a rule usually can not only produce a new fact, but also triggers an action based on certain conditions (a dedicated notation for such rules is, for example, ECA (Event-Condition-Action),
3. derivation – with backwards chaining, i.e. goal-oriented reasoning, which from a logical point of view is related to abduction,

4. integrity – wherein a set of rules defines conditions that have to be met for a certain system to operate properly,
5. constraint – similar to the above, wherein a set of rules can be transformed into a certain computational task to be solved; such rules are considered in CSP (constraint-satisfaction problems/programming),
6. association – such rules capture not decisions, but correlations between attributes, as is the case with rule induction in association rule mining in ML,
7. transformation – data and knowledge transformation procedures can be expressed with such rules, e.g. in the XSLT language (Clark 1999).

Moreover, they can be formalized on a different level. After (Ligeza 2006), the main levels of formalization to be identified are:

1) propositional – rules can be described using only propositional logic, and facts correspond to binary statements,
2) attributive – rule structure remains easy to interpret, but the language to describe facts is richer and allows numerous values and relations to be captured. This is the case of ALSV(FD) logic and XTT2 rules.
3) Horn – rules are logical clauses in first order logic, wherein all but one literals are true. Such rules are used in logic programming languages such as Datalog and Prolog, which also allow logical variables.
4) first order logic – rules are expressed with different subsets of first order logic; this is the case with the family of Description Logics (Baader 2003) and various other representations.

In fact, each of these representations can also impose different interpreter modes and applications.

Rule Interpretation Methods

Classic expert systems evolved into at least two different classes of systems. In the first are modules that can be integrated into a larger applications, as this is often the case with Jess. The second includes full-fledged frameworks such as Business Rules Management Systems (BRMS). Such systems provide complete platforms for developing decision support systems based on rules, including rule repositories and business process engines. A prime example of such system is Drools, which is composed of a number of modules, including an Expert rule engine, a Guvnor repository, and a jBPM process engine. In such a system, it is common to combine rules with a so-called business process that controls high-level inference flow.

In numerous works, we have proposed extensions of this model. In (Kluza 2012) we demonstrated the design of a process model combined with a rule model. Oryx, an open editor for BPMN, which is a visual notation for business processes, was enhanced to include XTT2 rule models. Furthermore, in (Kluza 2013) we proposed an algorithm that shows how a process model combined with rules could be designed. How such a model can be run was discussed in (Nalepa 2013). In that paper we proposed an extension of the open source process engine Activti that allows for integration with the HeaRT rule engine that runs the combined models.

Rules are still applied in a number of new domains; some are discussed in a recent book (Giurca 2009). One such application is the Semantic Web project (Berners-Lee 2001) that aimed to develop computer agents that automate common tasks for Internet users. The project introduced new important knowledge representation methods such as RDF and used formalized methods for reasoning, namely Description Logics (Baader 2003). Moreover, it assumed that reasoning would be performed with the use of rules. However, integrating typical production rules such as reasoning with the Semantic Web and Description Logics posed a major challenge.

A proposed solution to this problem was included in (Nalepa 2010), where we introduced a logical framework combining ALSV(FD) with Description Logics. This would allow for the combination of XTT2 rules with the formal ontologies used in the Semantic Web. Moreover, since a Web environment is distributed, many challenges arise with the design of the knowledge base. In fact, a number of users or agents generally cooperate in the design processes. Such a situation can be considered as collective knowledge engineering. In (Nalepa 2010) we considered this and proposed a semantic wiki system combined with a logical reasoning module in the Prolog language that supports the collaborative design process of the knowledge base.

Concluding Remarks

For many years, rules have been a useful symbolic knowledge representation in cognitive science and artificial intelligence. In this paper, we have discussed the common roots of the use of rules in these domains, i.e. cognitive architectures and production systems. We have characterized the main areas of study regarding rules in artificial intelligence. Furthermore, we have briefly commented on our contributions to rule-based systems. These include the ALSV(FD) logic, the XTT2 rule representation and design method, as well as their applications in business decision support systems, context-aware systems and the Semantic Web.

The development of rule-based systems has progressed much since the original production systems in the 1970s. While all the qualities of rules such as transparency and intuitive representation still hold, the design of large systems based of rules is a non-trivial task. It requires appropriate design methods and good quality analysis tools. Currently, mainstream AI seems to have little interest in using rules to model intelligence in a general sense. However, for AI researchers rules have become a standard yet still very elegant and effective engineering solution for modeling reasoning and decision support.

References

Agrawal, R., Imielinski, T., Swami, A. 1993. Mining Association Rules between Sets of Items in Large Databases. *Proceedings of the 1993 ACM Sigmod International Conference On Management Of Data*. Washington DC: 207–216.

Anderson, J.R. 1983. *The architecture of Cognition*. Cambridge, Mass.: Harvard University Press.

Anderson, J.R., *Rules of the mind*. 1993. Hillsdale, N.J.: Erlbaum.

Baader, F., Calvanese., D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.) 2003. *Description Logic Handbook*, Cambridge University Press.

Berners-Lee, T., Hendler, J., Ora, L. 2001. The Semantic Web. *Scientific American*, 284 (5).

Bobek, S., Nalepa, G.J. 2014. Incomplete and Uncertain Data Handling in Context-Aware Rule-Based Systems with Modified Certainty Factors Algebra, Rules on the Web. From Theory to Applications. *Lecture Notes in Computer Science*, 8620: 157–167.

Brachman, R., Levesque, H. 2004. *Knowledge Representation and Reasoning*, Morgan Kaufmann.

Brownston, L., Farrell, R., Kant, E., Martin, N. 1985. Programming Expert Systems in OPS5, Addison-Wesley.

Buchanan, B.G., Shortliffe, E.H. 1984. *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Reading, MA: Addison-Wesley.

Clark, J. 1999. XSL Transformations (XSLT) Version 1.0 W3C, World Wide Web Consortium (W3C).

Dey, A.K. 2000. *Providing architectural support for building context-aware applications*. Georgia Institute of Technology.

Domingos, P., Hulten, G. 2000. Mining High-speed Data Streams. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, Massachusetts: 71–80.

Flach, P. 2012. Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press.

Forgy, Ch., 1982. Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem. *Artificial Intelligence*, 1(19): 17–37.

Friedman-Hill, E. 2003. *Jess in Action: Rule Based Systems in Java*, Manning.

Gama, J. 2010. *Knowledge Discovery from Data Streams*. Chapman & Hall.

Giarratano, J.C., Riley, G.D. 2005. *Expert Systems*. Boston, MA: Thomson Course Technology.

Giurca, A., Gasevic, D., Taveter, K. (eds.) 2009. *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, New York, Hershey: Information Science Reference.

Hanson, E.N., Hasan, M.S. 1993. *Gator: An Optimized Discrimination Network for Active Database Rule Condition Testing*, CIS Department University of Florida.

Harmelen, F. van, Lifschitz, V., Porter, B. (eds.) 2007. *Handbook of Knowledge Representation*. Elsevier Science.

Hitzler, P., Kroetzsch, M., Rudolph, S. 2009. *Foundations of Semantic Web Technologies*. Chapman & Hall, CRC Press.

Klosgen, W., Żytkow, J.M. 2002. *Handbook of Data Mining and Knowledge Discovery*. New York: Oxford University Press.

Kluza, K., Kaczor, K., Nalepa, G.J. 2012. Enriching Business Processes with Rules using the Oryx BPMN Editor. Artificial Intelligence and Soft Computing: 11th International Conference, ICAISC 2012: Zakopane, Poland, April 29–May 3, 2012. *Lecture Notes in Artificial Intelligence*, 7268: 573–581.

Kluza, K., Nalepa, G.J. 2015. Generation of hierarchical business process models from Attribute Relationship Diagrams, Advances in ICT for business, industry and public sector: ABICT'13 (4th international workshop on Advances in business ICT): Krakow, September 8–11, 2013, Springer: 57–76.

Laird, J.E. 2012. *The Soar Cognitive Architecture*. MIT Press.

Liebowitz, J. 1998. *The Handbook of Applied Expert Systems*. Boca Raton, CRC Press.

Ligęza, A. 2006. *Logical Foundations for Rule-Based Systems*, Berlin, Heidelberg: Springer-Verlag.

Michalski, R.S. 1977. Synthesis of Optimal and Quasi-optimal Variable-valued Logic Formulas, *Proceedings of the 1975 International Symposium on Multiple Valued Logic*.

Miranker, D.P. 1987. *TREAT: A Better Match Algorithm for AI Production Systems*. University of Texas.

Nalepa, G.J. 2010. Architecture of the HeaRT Hybrid Rule Engine. Artificial Intelligence and Soft Computing: 10th International Conference, ICAISC 2010: Zakopane, Poland, June 13–17, 2010, Pt. II. Springer, *Lecture Notes in Artificial Intelligence*, 6114: 598–605.

Nalepa, G.J. 2010. Collective Knowledge Engineering with Semantic Wikis. *Journal of Universal Computer Science*, 7(16): 1006–1023.

Nalepa, G.J. 2011. *Semantic Knowledge Engineering. A Rule-Based Approach*, Kraków: Wydawnictwa AGH.

Nalepa, G.J., Bobek, S. 2014. Rule-Based Solution for Context-Aware Reasoning on Mobile Devices. *Computer Science and Information Systems*, 11(1): 171–193.

Nalepa, G.J., Bobek, S., Ligeza, A., Kaczor, K. 2011. Formalization and Modeling of Rules Using the XTT2 Method, International Journal on Artificial Intelligence Tools. *World Scientific*, 20(6): 1107–1125

Nalepa, G.J., Bobek, S., Ligeza, A., Kaczor, K. 2011. HalVA-Rule Analysis Framework for XTT2 Rules, Rule-Based Reasoning, Programming, and Applications. *Lecture Notes in Computer Science*, 6826: 337–344.

Nalepa, G.J., Furmanska, W.T. 2010. Integration Proposal for Description Logic and Attributive Logic – Towards Semantic Web Rules, Transactions on Computational Collective Intelligence II, *Lecture Notes in Computer Science*, 6450: 1–23.

Nalepa, G.J., Kluza, K., Kaczor, K. 2013. Proposal of an Inference Engine Architecture for Business Rules and Processes, Artificial Intelligence and Soft Computing: 12th International Conference, ICAISC 2013: Zakopane, Poland, June 9–13, 2013, *Lecture Notes in Artificial Intelligence*, 7895: 453–464.

Nalepa, G.J., Ligęza, A. 2010. HeKatE Methodology, Hybrid Engineering of Intelligent Systems, *International Journal of Applied Mathematics and Computer Science*, 20(1): 35–53.

Newell, A. 1990. *Unified theories of cognition*. Cambridge, Mass.: Harvard University Press.

Newell, A., Simon, H. 1972. *Human Problem Solving*. Englewood Cliffs, N.J.: Prentice Hall.

Quinlan, R. 1986. Induction of Decision Trees. *Machine Learning*, 1: 81–106, Boston: Kluwer Academic Publishers.

Vermesan, A.I., Coenen, F. (eds.) 1999. *Validation and Verification of Knowledge Based Systems: Theory, Tools and Practice*, Boston: Kluwer Academic Publisher.