J. Prentzas, I. Hatzilygeroudis, "Neurules – A Type of Neuro-Symbolic Rules: An Overview", in I. Hatzilygeroudis, J. Prentzas (Eds.), "Combinations of Intelligent Methods and Applications", Smart Innovation, Systems and Technologies, Vol. 8, pp. 145-165, ISBN: 978-3-642-19617-1, Springer Verlag, 2011.

Neurules-A Type of Neuro-Symbolic Rules: An Overview

Jim Prentzas[1], Ioannis Hatzilygeroudis[2]

Abstract Neurules are a kind of integrated rules integrating neurocomputing and production rules. Each neurule is represented as an adaline unit. Thus, the corresponding neurule base consists of a number of autonomous adaline units (neurules). Due to this fact, a modular and natural knowledge base is constructed, in contrast to existing connectionist knowledge bases. In this paper, we present an overview of our main work involving neurules. We focus on aspects concerning construction of neurules, efficient updates of neurule bases, neurule-based inference and combination of neurules with case-based reasoning. Neurules may be constructed from either symbolic rule bases or empirical data in the form of training examples. Due to the fact that the source knowledge of neurules may change with time, efficient updates of corresponding neurule bases to reflect such changes are performed. Furthermore, the neurule-based inference mechanism is interactive and more efficient than the inference mechanism used in connectionist expert systems. Finally, neurules can be naturally combined with case-based reasoning to provide a more effective representation scheme that exploits multiple knowledge sources and provides enhanced reasoning capabilities.

1 Introduction

The combination or integration of (two or more) different problem solving methods has given fruitful results in many application areas. The aim is to create combined formalisms or systems that benefit from each of their components. Disadvantages or limitations of specific intelligent methods can be surpassed or

[1] Democritus University of Thrace, School of Education Sciences, Dept. of Education Sciences in Pre-School Age, Nea Chili, 68100 Alexandroupolis, Greece. Email: dprentza@psed.duth.gr

[2] University of Patras, School of Engineering, Dept. of Computer Engineering and Informatics, 26500 Patras, Greece. Email: ihatz@ceid.upatras.gr

alleviated by their combination with other methods. It is generally believed that complex problems can be easier solved with such combinations (Medsker 1995).

A popular type of combinations is that of symbolic and connectionist approaches, usually called the neuro-symbolic approach. Advanced neuro-symbolic formalisms and systems have been developed (Bookman and Sun 1993, Fu 1994, Medsker 1995, Hilario 1997, Sun and Alexandre 1997, McGarry et al. 1999; Wermter and Sun 2000, Cloete and Zurada 2000, d'Avila Garcez et al 2002, d'Avila Garcez et al 2004, Hatzilygeroudis and Prentzas 2004a, Bader and Hitzler 2005). Different types of neuro-symbolic approaches have been developed such as combinations of connectionist approaches with first-order logic (Bader et al. 2008, Shastri 2007), or with multi-valued logic (Komendantskaya et al. 2007) or with non-classical logic (d'Avila Garcez et al. 2007) or with symbolic rules (of propositional type) (Gallant 1993, Towell and Shavlik 1994, Fu 1993, Hatzilygeroudis and Prentzas 2000b and 2001b). However, combinations of neural networks and symbolic rules seem to have given more applied results (Souici-Meslati and Sellami 2006, Xianyu et al. 2008, Yu et al. 2008) due to the complementary advantages and disadvantages of the two combined formalisms (Hatzilygeroudis and Prentzas 2004a).

Symbolic rules have several advantages as well as some significant disadvantages in terms of knowledge representation and reasoning. Their main advantages involve naturalness of representation and modularity (see e.g. Reichgelt 1991). The naturalness of rules facilitates comprehension of their encompassed knowledge. Modularity refers to the fact that each rule is a discrete, autonomous unit enabling incremental development of the knowledge base as well as partial testing. Moreover, rule based systems provide an interactive inference mechanism, which guides the user in supplying input values, and an explanation mechanism, which justifies the reached conclusions. The provision of explanations is necessary in certain application domains (e.g. medicine) to justify system outputs. Symbolic rules have certain drawbacks besides advantages. An important disadvantage concerns the knowledge acquisition bottleneck that is, the difficulty in acquiring rules from experts (see e.g. Gonzalez and Dankel 1993). The brittleness of rules is another disadvantage. More specifically, it is not possible to draw conclusions from rules when there are missing values in the input data. For a specific rule, a certain number of condition values must be known in order to evaluate the logical function connecting its conditions. In addition, rules do not perform well in cases of unexpected input values or combinations of them.

Neural networks represent a totally different approach to problem solving, known as connectionism (see e.g. Gallant 1993, Haykin 2008). Neural networks possess certain advantages but disadvantages as well. They are able to obtain knowledge from training examples. Therefore, empirical knowledge (i.e. training examples) available in several domains is exploited and interaction with the experts is reduced. Additional advantages of neural networks concern their ability to generalize that is, provide computation of correct outputs from input combinations not present in the training set, their ability to represent complex and imprecise

knowledge and their efficiency in producing outputs. Compared to symbolic rules, neural networks possess significant disadvantages. Main such disadvantages concern the lack of naturalness and modularity. It is difficult to comprehend the knowledge encompassed in neural networks and for this reason several rule extraction methods have been presented (Andrews et al. 1995). Due to the lack of modularity, a neural network cannot be decomposed into components and form a modular structure. The aforementioned drawbacks result into the difficulty (if not inability) in providing explanations for outputs produced by neural networks.

From the various neuro-symbolic approaches that have been presented, we concentrate on combinations that result in a uniform, seamless combination of the two component approaches. Such combinations are called unified, according to (Hilario, 1997), or integrated, according to (Bader and Hitzler, 2005). A main research direction at combining rules and neural networks involves use of prior domain knowledge in neural network configuration. One could discern two different trends in this research direction. The one trend stems from (Holldobler and Kalinke 1994), where a connectionist network is developed that implements the meaning function of a propositional (definite) logic program. The other trend stems from (Towell and Shavlik 1994), which consists of two main steps: an existing domain theory in the form of propositional rules is used to construct an initial neural network and then training data are used to train the network. On the other hand, connectionist expert systems are integrated systems that represent relationships between concepts associated with nodes in a neural network (Gallant 1988, Gallant 1993, Ghalwash 1998). The network also contains certain random cells that have no concepts assigned to them. These cells are introduced during construction.

Most (if not all) of existing such approaches give pre-eminence to connectionism. Thus, they do not exploit representational advantages of symbolic rules, like naturalness and modularity. Moreover, with the exception of connectionist expert systems, they do not provide the functionalities of a rule-based system, like interactive inference and explanation. It should also be mentioned that as far as connectionist expert systems are concerned, the presence of random cells results in certain incomprehensible explanations.

Neurules (Hatzilygeroudis and Prentzas 2000a, Hatzilygeroudis and Prentzas 2000b, Hatzilygeroudis and Prentzas 2001b) are a type of integrated rules combining symbolic rules (of propositional type) and neurocomputing. In contrast to other approaches, neurules give pre-eminence to the symbolic part of the integration. Therefore, they retain the naturalness and modularity of symbolic rules in a large degree. Neurules can be produced either from symbolic rules or from empirical data (Hatzilygeroudis and Prentzas 2000a, 2001b). Also a neurule-based system possesses an interactive inference mechanism (Hatzilygeroudis and Prentzas 2010) and provides explanations for drawn conclusions (Hatzilygeroudis and Prentzas 2001a). Mechanisms for efficiently updating a neurule base, given changes to its source knowledge (i.e. symbolic rules or empirical data), have also been developed (Prentzas and Hatzilygeroudis 2005, Prentzas and Hatzilygeroudis 2007b).

Neurules may also be effectively combined with case-based reasoning (Prentzas and Hatzilygeroudis 2002, Hatzilygeroudis and Prentzas 2004c).

In this paper, we present an overview of our work concerning neurules. The structure of the paper is as follows. Section 2 presents the neurule-based knowledge representation scheme. In Section 3 production of neurules from existing symbolic rules is presented. Section 4 discusses aspects regarding the mechanism for efficiently updating a neurule base given changes to its symbolic source knowledge (i.e. symbolic rule base). Section 5 outlines construction of neurules from empirical data. Section 6 briefly discusses aspects regarding efficient updates of a neurule base due to availability of new empirical source data. Section 7 discusses the interactive neurule-based inference mechanism. Section 8 discusses issues concerning combination of neurules with case-based reasoning. Finally, Section 9 concludes.

2 Neurules

2.1 Syntax and Semantics

Neurules are a kind of integrated rules. The form of a neurule is depicted in Fig.1a. Each condition C_i is assigned a number sf_i, called its *significance factor*. Moreover, each rule itself is assigned a number sf_0, called its bias factor. Internally, each neurule is considered as an adaline unit (Fig.1b). The inputs C_i (i=1,...,n) of the unit are the conditions of the rule. The weights of the unit are the significance factors of the neurule and its bias is the bias factor of the neurule. Each input takes a value from the following set of discrete values: [1 (true), -1 (false), 0 (unknown)].

The output D, which represents the conclusion (decision) of the rule, is calculated via the standard formulas:

$$D = f(a), a = sf_0 + \sum_{i=1}^{n} sf_i C_i \tag{1}$$

$$f(a) = \begin{cases} 1, & if \quad a \geq 0 \\ -1, & if \quad a < 0 \end{cases} \tag{2}$$

where ***a*** is the *activation value* and *f(x)* the *activation function*, which is a threshold function. Hence, the output can take one of two values (‘-1’, ‘1’) representing failure and success of the rule respectively. The significance factor of a condition represents the significance (weight) of the condition in drawing the con-

clusion. The LMS learning algorithm is used to compute the values of the significance factors as well as the bias factor of a neurule. Examples of neurules are shown in Table 3.

The general syntax of a neurule (in a BNF notation, where '< >' denotes non-terminal symbols) is:

<rule>::= (<bias-factor>) **if** <conditions> **then** <conclusion>
<conditions>::= <condition> | <condition>,<conditions>
<condition>::= <variable> <l-predicate> <value> (<significance-factor>)
<conclusion>::= <variable> <r-predicate> <value> .

where <variable> denotes a *variable*, that is a symbol representing a concept in the domain, e.g. 'sex', 'pain' etc in a medical domain, and <l-predicate> denotes a symbolic or a numeric predicate. The symbolic predicates are {is, isnot}, whereas the numeric predicates are {<, >, =}. <r-predicate> can only be a symbolic predicate. <value> denotes a value; it can be a symbol (e.g. "male", "night-pain") or a number (e.g "5"). <bias-factor> and <significance-factor> are (real) numbers.

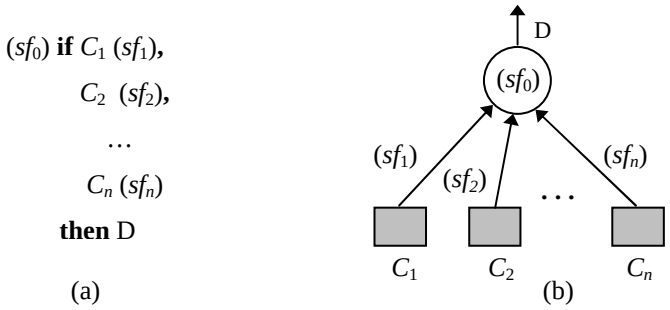

Fig. 1. (a) Form of a neurule (b) a neurule as an adaline unit

We distinguish three types of variables:

- *input or askable variables*, that is variables for which the user will be prompted to give a value during inference,
- *intermediate or inferable variables*, that is variables constituting intermediate goals of the inference process,
- *output or goal variables*, that is variables constituting the (final) goals of the inference process.

We also distinguish between *input*, *intermediate* and *output neurules*. An input neurule is a neurule having only input variables in its conditions and intermediate or output variables in its conclusions. An intermediate neurule is a neurule having at least one intermediate variable in its conditions and intermediate variables in its conclusions. An output neurule is one having an output variable in its conclusions.

2.2 Neurule-Based System Architecture

In Figure 2, the architecture of a neurule-based system is illustrated. The run-time system (in the dashed rectangle) consists of five modules: the *neurule base (NRB)*, the *hybrid inference engine (HIE)*, the *working memory (WM)*, the *explanation mechanism (EXM)* and the *indexed case library (ICL)*. The first four of these modules are more or less functionally similar to those of a conventional rule-based system. HIE in combination with ICL can provide additional reasoning capabilities (i.e. handling of exceptional situations).

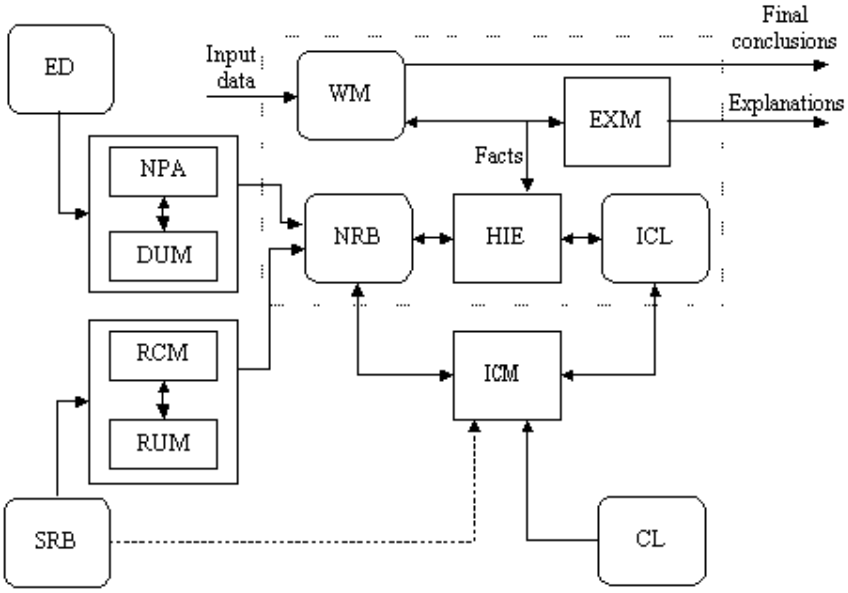

Fig. 2. The architecture of a neurule-based system

NRB contains neurules alongside certain information useful for updating neurules when the source knowledge changes (see Sections 4 and 6). HIE is responsible for making inferences. HIE either performs purely neurule-based inference by taking into account the data in WM and the neurules in NRB or combines neurule-based with case-based reasoning by also taking into account cases stored in the ICL. WM contains *fact assertions* either given by the user, as initial input data or during an inference course, or produced by the system, as intermediate or final conclusions during an inference course. ICL contains cases indexed by neurules in the NRB and is used by the approach combining neurule-based with case-based reasoning (see Section 8).

The architecture also includes certain offline modules useful for producing and updating the contents of the NRB and for constructing an *indexed case library (ICL)*. The contents of the NRB are produced from a *symbolic rule base (SRB)* or from *empirical data (ED)*. Construction of a NRB from a symbolic rule base is

performed by the *rule conversion mechanism (RCM)* presented in Section 3. Construction of a NRB from empirical data is performed by the neurules production algorithm (NPA) presented in Section 5. The *rule update mechanism (RUM)* updates the NRB to reflect changes to its symbolic rule source (see Section 4). RUM interacts with the RCM to perform its tasks. The *data update mechanism (DUM)* updates the NRB when new empirical data becomes available (see Section 6). The *indexing construction mechanism (ICM)* constructs an ICL by taking as input a *case library (CL)* and either symbolic rules indexing cases in CL or neurules.

3 Construction of a Neurule Base from a Symbolic Rule Base

As mentioned above, neurules can be produced from either symbolic rules or empirical data. Here, we concentrate on the former.

An existing (propositional type) SRB can be converted to a neurule base (NRB) by the rule conversion mechanism. The corresponding conversion mechanism is described in (Hatzilygeroudis and Prentzas 2000b). Conversion does not involve refinement of SRB, but creates an equivalent knowledge base. This means that the conclusions drawn from NRB are the same as those drawn from SRB, given the same inputs. Each produced neurule usually merges two or more symbolic rules with the same conclusion. Therefore, the size of the produced NRB is less than that of SRB as far as both the number of rules and the number of conditions is concerned. This results in improvements to the efficiency of the inferences from NRB, compared to those from SRB (Hatzilygeroudis and Prentzas 2000b).

The conversion mechanism is outlined as follows:

1. Group symbolic rules into merger sets.
2. From each merger set, produce a merger.
3. Produce a training set for each merger.
4. Train each merger and produce one or more neurules.

Each *merger set* contains all the rules of the SRB having the same conclusion. We call such merger sets *initial merger sets*. A *merger* is a neurule having as conditions all the conditions of the symbolic rules in the corresponding merger set (without duplications) and significance factors as well as bias factor set to zero (or any other proper initial value). Each training set is extracted from the truth table of the combined logical function of the rules in its merger set (the disjunction of the conjunctions of the conditions of each rule), via a filtering process. Filtering eliminates the invalid rows of the truth table. Invalid rows are those with contradicting or inconsistent values.

Training of mergers is performed using the standard LMS algorithm. A limitation of the LMS algorithm is its inability to find a set of significance and bias factors that classify correctly all of the training patterns, in case that the training patterns of the training set are inseparable. In case that training is successful, one

neurule will be produced. Otherwise, a splitting process is followed, which produces more than one neurule having the same conclusion, called sibling neurules.

Table 1. An example merger set

R1	**R5**
if patient-class is human0-20, pain-feature2 is continuous, fever is no-fever, antinflam-reaction is none	**if** patient-class is human0-20, pain-feature2 is night, fever is no-fever, antinflam-reaction is none
then disease-type is primary-malignant	**then** disease-type is primary-malignant
R2	**R6**
if patient-class is human0-20, pain-feature2 is night, fever is low	**if** patient-class is human21-35, pain-feature2 is night, antinflam-reaction is none
then disease-type is primary-malignant	**then** disease-type is primary-malignant
R3	**R7**
if patient-class is human0-20, pain-feature2 is night, fever is medium	**if** patient-class is human36-55, pain-feature2 is night, fever is low
then disease-type is primary-malignant	**then** disease-type is primary-malignant
R4	**R8**
if patient-class is human0-20, pain-feature2 is night, fever is high	**if** patient-class is human36-55, pain-feature2 is night, fever is medium
then disease-type is primary-malignant	**then** disease-type is primary-malignant

Splitting is performed in a way that each subset contains symbolic rules that are 'close' to each other in some degree. *Closeness* between two symbolic rules is defined as the number of their common conditions. Splitting is based on the notion of closeness due to the observation that separable sets have rules with larger average closeness than inseparable ones. A *least closeness pair (LCP)* of rules in a merger set is a pair of rules with the least closeness (LC) in the set. Initially, a LCP in the merger set is found and two subsets are created each containing as its initial element one of the rules of that pair, called its pivot. Each of the remaining rules is distributed between the two subsets based on their closeness to their pivots. That is, each subset contains rules, which are closer to its pivot. If training fails, for a merger of a merger subset, the corresponding subset is further split into

two other subsets, based on one of its LCPs. This continues, until training succeeds or the merger subset contains only one rule that is converted into a neurule.

As an example, to demonstrate application of the main steps of the conversion mechanism, we use the merger set of Table 1 that consists of eight symbolic rules {R1, R2, R3, R4, R5, R6, R7, R8}, taken from a medical diagnosis rule base. The merger constructed from this (initial) merger set contains the ten distinct conditions of the eight rules and is shown in Table 2. The training set of the merger is extracted from the truth table of the combined logical function of the rules of the merger set:

$$F = (C1 \wedge C2 \wedge C3 \wedge C4) \vee (C1 \wedge C5 \wedge C6) \vee (C1 \wedge C5 \wedge C7) \vee (C1 \wedge C5 \wedge C8) \vee (C1 \wedge C5 \wedge C3 \wedge C4) \vee (C9 \wedge C5 \wedge C4) \vee (C10 \wedge C5 \wedge C6) \vee (C10 \wedge C5 \wedge C7)$$

where C1≡patient-class is human0-20, C2≡pain-feature2 is continuous, C3≡fever is no-fever, C4≡antinflam-reaction is none, C5≡pain-feature2 is night, C6≡ fever is low, C7≡ fever is medium, C8≡ fever is high, C9≡ patient-class is human21-35, C10≡ patient-class is human36-55.

Table 2. The merger of the merger set of Table 1

(0) **if** patient-class is human0-20 (0),
pain-feature2 is continuous (0),
fever is no-fever (0),
antinflam-reaction is none (0),
pain-feature2 is night (0),
fever is low (0),
fever is medium (0),
fever is high (0),
patient-class is human21-35 (0),
patient-class is human36-55 (0)
then disease-type is primary-malignant

The truth table of F contains 2^{10}=1024 training patterns, from which only 120 patterns remain after application of the filtering process. The training patterns of the training set are inseparable and the initial merger set is split in two subsets: MS1={R1, R5, R6} and MS2={R2, R3, R4, R7, R8}. The LCP that guides splitting is (R1, R7). Training of the merger of MS1 is not successful. So, {R1, R5, R6} is split in {R1, R5} and {R6} with LCP: (R1, R6). The merger of {R1, R5} is successfully trained and neurule NR1-5 is produced. Rule R6 is converted to a neurule (i.e. NR6). The merger of MS2 is successfully trained and neurule NR2-3-4-7-8 is produced. So, finally, from the initial merger set of eight symbolic rules, three neurules are produced. The produced neurules are shown in Table 3.

Table 3. Neurules produced from the merger set of Table 1

NR1-5	**NR2-3-4-7-8**
(-2.5) **if** fever is no-fever (1.4), antinflam-reaction is none (1.3), patient-class is human0-20 (0.8), pain-feature2 is continuous (0.8), pain-feature2 is night (0.8) **then** disease-type is primary-malignant	(1.6) **if** patient-class is human0-20 (8.5), pain-feature2 is night (8.2), fever is medium (8.2), patient-class is human36-55 (5.0), fever is low (4.4), fever is high (0.8) **then** disease-type is primary-malignant
NR6	
(-2.4) **if** patient-class is human21-35 (1.5), pain-feature2 is night (1.4), antinflam-reaction is none (1.3) **then** disease-type is primary-malignant	

4 Efficient Updating of a Neurule Base Produced from a Symbolic Rule Base

An aspect of interest involves the efficient updates of a NRB to reflect changes to its symbolic source knowledge. The basic changes to SRB can be (a) insertion of a new rule and (b) removal (or deletion) of an existing rule, since modification of a rule is equivalent to removal of the old rule and insertion of the new one. One approach to reflect such changes would be to reconvert the whole SRB and reproduce the whole NRB. Obviously such an approach would impose useless computational effort due to the fact that only specific parts of the SRB are affected from changes. To minimize the computational effort for performing updates, an efficient mechanism has been developed (Prentzas and Hatzilygeroudis 2005) that reconverts as small portion of SRB as possible. The modularity of NRB enables such an approach. Furthermore, the number of neurules after an update remains as small as possible, which is a significant aspect in terms of inference time-efficiency.

The update mechanism exploits the structure of a tree, called the splitting tree, that stores information related to the conversion process. More specifically, a splitting tree is used to store the splitting process for each initial merger set. The root of a tree corresponds to an initial merger set. The intermediate nodes and leaves correspond to the subsequent subsets, into which the initial merger set was split. An intermediate node denotes a subset that was split, due to training failure, whereas a leaf denotes a subset that was successfully trained and produced a neurule. The pivot of each (sub)set is attached to the corresponding branch of the tree.

Figure 3 depicts the splitting tree corresponding to the splitting process for the merger set of Table 1.

Whenever a new symbolic rule R is inserted in SRB and there are more than one sibling neurule of R in NRB, the splitting tree is exploited to focus the update process on the neurules produced from the merger subset containing the rules closest to R. To achieve this, the splitting tree is traversed, starting from the root. Traversing is based on the closeness of the inserted rule to the LCP members of the merger subsets corresponding to the traversed nodes. Traversing ends at an intermediate node, when the corresponding merger subset contains a rule R' whose closeness to the inserted rule is less than the least closeness. Otherwise, traversing ends at a leaf. In case traversing stops at a leaf, the corresponding neurule is removed from the NRB, the merger corresponding to the new merger set is trained, updating accordingly the NRB and the splitting tree. In case traversing stops at an intermediate node, the descending nodes as well as corresponding neurules are removed, the new merger set is split in two subsets based on LCP (R, R'), the two corresponding mergers are trained updating accordingly the NRB and the splitting tree. In any case, parts of the initial splitting tree are exploited to avoid useless computational effort.

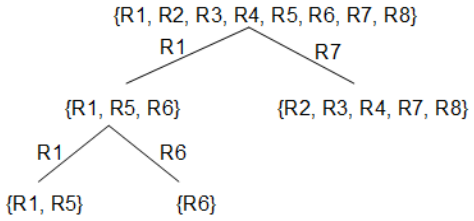

Fig. 3. The splitting tree for merger set of Table 1

Table 4. Inserted rule R9 and resulted neurule NR2-3-4-7-8-9

R9	**NR2-3-4-7-8-9**
if patient-class is human36-55, pain-feature2 is night, fever is high **then** disease-type is primary-malignant	(1.6) **if** pain-feature2 is night (8.2), fever is high (8.0), patient-class is human36-55 (5.0), patient-class is human0-20 (4.9), fever is medium (4.6), fever is low (4.4) **then** disease-type is primary-malignant

Whenever an existing symbolic rule R is removed from SRB and there are more than one sibling neurule of R in NRB, the splitting tree is exploited in a way similar to the approach for rule insertion. Traversing ends at an intermediate node,

when R is a member of LCP of its merger (sub)set. Otherwise, traversing ends at a leaf. Each case is handled accordingly.

It should be mentioned that in certain situations of rule insertion/removal, the number of neurules contained in the NRB may decrease by one. A detailed presentation of the update mechanism along with experimental results is presented in (Prentzas and Hatzilygeroudis 2005).

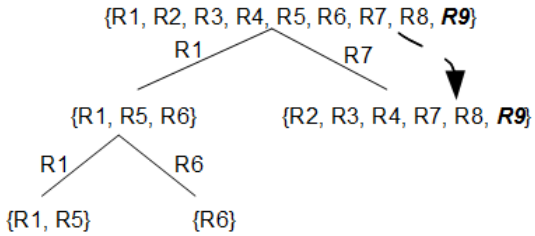

Fig. 4. Insertion of R9: traversal of the splitting tree

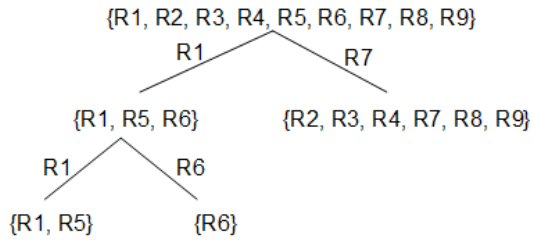

Fig. 5. Final form of the splitting tree after insertion of R9

We will demonstrate application of the update mechanism for rule insertion with an example. Let us consider the rules in Table 1 as constituting SRB and those in Table 3 as constituting NRB. Also, suppose that rule R9 (Table 4) is to be inserted. Information contained in the splitting tree shown in Figure 3 is exploited to efficiently perform the update of the NRB. Traversing of the splitting tree ends at the leaf related to subset {R2, R3, R4, R7, R8} (Figure 4). Notice that R9 is inserted into the merger sets corresponding to all traversed nodes. NR2-3-4-7-8 is removed from NRB. Training of the merger corresponding to the new merger (sub)set {R2, R3, R4, R7, R8, R9} is successful and the corresponding neurule NR2-3-4-7-8-9 is inserted into NRB (Table 4). The splitting tree takes the form shown in Figure 5.

5 Producing a Neurule Base from Empirical Data

In several domains, empirical data in the form of training examples are available and can be exploited to construct neurule bases. The neurules production algorithm (NPA) constructs a neurule base from empirical data. NPA requires the following input: a set of domain variables V representing the domain concepts with their possible values, possible dependency information among domain variables and a set of empirical data S. Dependency information indicates which variables the intermediate, if any, and output variables depend on.

NPA tries to produce one neurule for each output/intermediate variable value that is, one neurule for each possible output/intermediate conclusion. This is not always possible due to the fact that the training set may be inseparable. Therefore, more than one neurule having the same conclusion may be produced (i.e. *sibling neurules*). The main steps of NPA are outlined as follows:

1. Construct *initial neurules*, based on dependency information.
2. Extract an *initial training set* for each initial neurule from S.
3. Train each initial neurule individually and produce corresponding neurule(s).

Initial neurules represent the possible intermediate or final conclusions. One initial neurule is constructed for each value of each intermediate or output variable. The conditions of each initial neurule include the variables that contribute in drawing the corresponding conclusion, as specified by the dependency information. Then, for each initial neurule its corresponding initial training set is extracted from the empirical dataset. A training pattern has the form $[v_1\ v_2\ \ldots\ v_n\ d]$, where d is the desired value of a variable related to an intermediate or output conclusion and v_i, i=1,…,n are the values of the variables it depends on, called component values. We distinguish between *success examples* and *failure examples* in a training set. Success examples are those having '1' ('true') as their desired value, whereas failure examples are those having '-1' ('false'). Each initial neurule is individually trained, via the Least Mean Square (LMS) algorithm, using its own training set. Training is not always successful, that is a set of significance and bias factors cannot always be found that correctly classify all of the training examples. This is the case when the training patterns are inseparable. When the algorithm succeeds, that is values for the bias and significance factors are calculated that classify all training patterns, one neurule is produced. When it fails, due to inseparability of the training examples, a splitting process is followed. More specifically, the initial training set of the neurule is split into two subsets and two copies of the initial neurule are trained, each using one of the training subsets. If training of either neurule copy fails, its subset is further split into two other subsets and so on, until there is no failure or a subset contains only one success pattern. In this way, more than one neurule are produced, having the same conditions with different bias and significance factors and the same conclusion.

Splitting is based on the notion of closeness between training patterns. The closeness between two examples is defined as the number of their common component values. A *least closeness pair (LCP)* consists of two success examples that have the least closeness between them. Splitting a training set is based on an LCP. More specifically, each subset comprises one of the members of an LCP, the success examples closer to it and all the failure examples of the initial training set. This stems from the intuition that existence of quite different examples causes inseparability.

Table 5. The empirical data set S for the lenses example problem

age	spectacle	astigmatic	tear-rate	lenses-class
young	myope	no	reduced	no-lenses
young	myope	no	normal	soft-lenses
young	myope	yes	reduced	no-lenses
young	myope	yes	normal	hard-lenses
young	hypermetrope	no	reduced	no-lenses
young	hypermetrope	no	normal	soft-lenses
young	hypermetrope	yes	reduced	no-lenses
young	hypermetrope	yes	normal	hard-lenses
pre-presbyopic	myope	no	reduced	no-lenses
pre-presbyopic	myope	no	normal	soft-lenses
pre-presbyopic	myope	yes	reduced	no-lenses
pre-presbyopic	myope	yes	normal	hard-lenses
pre-presbyopic	hypermetrope	no	reduced	no-lenses
pre-presbyopic	hypermetrope	no	normal	soft-lenses
pre-presbyopic	hypermetrope	yes	reduced	no-lenses
pre-presbyopic	hypermetrope	yes	normal	no-lenses
presbyopic	myope	no	reduced	no-lenses
presbyopic	myope	no	normal	no-lenses
presbyopic	myope	yes	reduced	no-lenses
presbyopic	myope	yes	normal	hard-lenses
presbyopic	hypermetrope	no	reduced	no-lenses
presbyopic	hypermetrope	no	normal	soft-lenses
presbyopic	hypermetrope	yes	reduced	no-lenses
presbyopic	hypermetrope	yes	normal	no-lenses

To demonstrate application of NPA, we use an example problem taken from the UCI Machine Learning ftp repository (Frank and Asuncion 2010); it is called the LENSES problem. There are five domain variables, four input (i.e. age, spectacle, astigmatic, tear-rate) and one output (lenses-class) that depends on the four input variables. Table 5 shows the corresponding empirical dataset consisting of

twenty-four (24) patterns. The output variable takes three possible values (i.e. no-lenses, soft-lenses, hard-lenses) and therefore three initial neurules are constructed. A training set is extracted for each initial neurule. Each of the three training sets consists of twenty-four (24) patterns. The patterns in each of them have the same input values, but different output values.

The (final) neurules produced are shown in Table 6. For the first two initial neurules, the calculated factors successfully classified all training patterns. The produced neurules NR1 and NR2. However, the same didn't happen with the third initial neurule. Its training set had to be split, two copies of the third initial neurule were trained with each subset and neurules and finally neurules NR3 and NR4 were produced.

Table 6. The neurules produced from the empirical set of lenses problem

NR1	**NR**3
(-13.1) **if** age is young (8.8), age is pre-presbyopic (1.5), age is presbyopic (1.2), spectacle is myope (1.6), spectacle is hypermetrope (-2.7), astigmatic is no (-6.1), astigmatic is yes (4.4), tear-rate is reduced (-5.7), tear-rate is normal (4.6) **then** lenses-class is hard-lenses	(-4.6) **if** age is young (-4.4), age is pre-presbyopic (2.6), age is presbyopic (3.2), spectacle is myope (-4.2), spectacle is hypermetrope (3.4), astigmatic is no (-4.5), astigmatic is yes (3.3), tear-rate is reduced (6.5), tear-rate is normal (-8.0) **then** lenses-class is no-lenses
NR2	**NR**4
(-14.6) if age is young (6.4), age is pre-presbyopic (6.9), age is presbyopic (-0.4), spectacle is myope (-3.9), spectacle is hypermetrope (3.1), astigmatic is no (6.9), astigmatic is yes (-7.4), tear-rate is reduced (-7.9), tear-rate is normal (6.2) **then** lenses-class is soft-lenses	(-2.2) if age is young (-2.6), age is pre-presbyopic (-2.5), age is presbyopic (5.0), spectacle is myope (1.0), spectacle is hypermetrope (-2.5), astigmatic is no (5.1), astigmatic is yes (-6.2), tear-rate is reduced (8.1), tear-rate is normal (-9.5) **then** lenses-class is no-lenses

6 Efficient Updating of a Neurule Base Produced from Empirical Data

In certain domains, training examples become available over time. Therefore, an aspect of interest involves the efficient updates of a NRB to reflect availability of new empirical source knowledge. In (Prentzas and Hatzilygeroudis 2007b) we present an efficient mechanism for performing such updates. The mechanism is based on splitting trees containing information regarding the splitting process for each training set of each initial neurule, in a similar way to that in Section 4. The root of each tree corresponds to the initial training set. Descendant nodes correspond to the subsequent subsets into which the initial training set was split. Each leaf denotes subsets that was successfully trained and produced a neurule. The members of the LCP that guided each split are attached to the corresponding branches of the tree.

The splitting tree is useful to perform updates in case more than one sibling neurules have been produced. The availability of a new training example means insertion of a new success example into a specific initial training set and insertion of a new failure example into all other initial training sets. Splitting trees enable to perform such updates efficiently.

To insert a new success example not satisfied by the existing neurules produced from the initial training set, the corresponding splitting tree is traversed to starting from the root and ending at a leaf or an intermediate node. Traversing is based on the closeness between the new success example and the LCPs attached to the edges of the splitting tree. Retraining of the corresponding subset is performed updating the NRB and the splitting tree.

The insertion of a failure example not satisfied by the existing neurules into an initial training set requires training of the subsets corresponding to leaves of the splitting tree whose corresponding neurules misclassify the new failure example. The corresponding existing neurules are removed from the NRB, whereas the newly created ones are inserted.

7 Neurule-based Inference Engine

The *neurule-based inference engine* implements the way neurules co-operate to reach a conclusion. The choice of the next rule to be considered is based on a neurocomputing measure, but the rest is symbolic (Hatzilygeroudis and Prentzas 2010).

Generally, the output of a neurule is computed according to Eq. (1) (Section 2.1). However, it is possible to deduce the output of a neurule without knowing the values of all of its conditions. To achieve this, we define for each neurule the *known sum (kn-sum)* and the *remaining sum (rem-sum)*. More specifically, ‘known-sum’ is the weighted sum of the values of the already known (i.e. evalu-

ated) conditions (inputs) of the corresponding neurule. 'Remaining sum' is the sum of absolute values of significance factors corresponding to all unevaluated conditions of the neurule. Therefore, 'remaining sum' represents the largest possible weighted sum of the remaining (i.e. unevaluated) conditions of the neurule.

If $|kn\text{-}sum| > rem\text{-}sum$ for a certain neurule, then evaluation of its conditions can stop, because its output can be deduced regardless of the values of the unevaluated conditions. In this case, its output is guaranteed to be '1' if $kn\text{-}sum > 0$ whereas it is '-1', if $kn\text{-}sum < 0$. In the first case, we say that the neurule is *fired*, whereas in the second that it is *blocked*.

A condition evaluates to 'true', if it matches a fact in the working memory that is, there is a fact with the same variable, predicate and value. A condition evaluates to 'unknown', if there is a fact with the same variable, predicate and 'unknown' as its value. A condition cannot be evaluated if there is no fact in the working memory with the same variable. In this case, either a question is made to the user to provide data for the variable, in case of an input variable, or an intermediate neurule with a conclusion containing the variable is examined, in case of an intermediate variable. A condition with an input variable evaluates to 'false', if there is a fact in the working memory with the same variable, predicate and different value. A condition with an intermediate variable evaluates to 'false' if additionally to the latter there is no unevaluated intermediate neurule that has a conclusion with the same variable. Inference stops either when one or more output neurules are fired (success) or there is no further action (failure). To facilitate inference, conditions of neurules are organized according to the descending order of their significance factors.

In (Hatzilygeroudis and Prentzas 2001a) we present initial work for the provision of explanations concerning neurule-based inference. Explanations involve 'how' type rules justifying how conclusions were reached.

Neurule-based inference has certain advantages. When a neurule base is produced from a symbolic rule base, experimental results have shown that neurule-based inference is more efficient than the corresponding symbolic rule-based inference (Hatzilygeroudis and Prentzas 2000b). Another advantage of neurule-based reasoning compared to symbolic rule-based reasoning is the ability to reach conclusions from neurules even if some of the conditions are unknown. This is not possible in symbolic rule-based reasoning. A symbolic rule needs all its conditions to be known in order to produce a conclusion.

Most neuro-symbolic approaches, except connectionist expert systems, do not support functionalities like interactive inference and provision of natural explanations. Neurule-based inference is more efficient than the inference mechanism used in connectionist expert systems (Hatzilygeroudis and Prentzas 2010).

8 Combining Neurule-Based and Case-Based Reasoning

Case-based reasoning is an approach that exploits knowledge encompassed in stored past cases to handle similar new cases (Aamodt and Plaza 1994). It is useful in several domains where an abundant number of past cases are available. A case-based system stores useful experience obtained when handling new cases and is continuously enhanced during operation. Case-based representations offer several advantages such as easy knowledge acquisition, naturalness, modularity, ability to express specialized knowledge, self-updatability. There are also issues of CBR that may give rise to problems such as adaptation, inference efficiency regarding case retrieval, provision of explanations, difficulties in knowledge acquisition in certain domains (Prentzas and Hatzilygeroudis 2007a, 2009).

Combinations of case-based reasoning with other intelligent methods have been pursued in several domains resulting into more effective representation schemes. One of the most effective types of combinations involves combination of case-based reasoning with rule-based reasoning (Prentzas and Hatzilygeroudis 2007a). Such combinations offer benefits since the advantages of rule-based reasoning and case-based reasoning are complementary to a large degree. An overall advantage of such combined approaches involves naturalness and modularity of the representation scheme. Neurules are a type of integrated rules offering advantages compared to symbolic rules. In (Prentzas and Hatzilygeroudis 2002, Hatzilygeroudis and Prentzas 2004c) we explored the combination of neurule-based with case-based reasoning. A main benefit, among others, deriving from this combination concerns accuracy improvement as cases may fill in gaps of neurules in domain knowledge representation. Furthermore, characteristics of the formalism involve naturalness, modularity, ability to exploit multiple types of knowledge sources and self-updatability. Few approaches combine case-based reasoning with multiple other intelligent methods with the other methods being outside the case-based reasoning cycle.

In the representation scheme combining neurules with case-based reasoning, neurules index cases representing their exceptions. The indexing construction module implements the process of acquiring an indexing scheme. The specific process may take as input alternative types of available knowledge: (a) available neurules and cases or (b) available symbolic rules and exception cases.

Let us consider that the indexing process takes as input available neurules and cases. To acquire an indexing scheme, neurule-based reasoning is performed for the neurules based on the attribute values of each case. A case is indexed as a neurule's exception, whenever the neurule fires and the value of the conclusion variable do not match the corresponding attribute value of the case.

The alternative type of knowledge concerns an available formalism of symbolic rules and indexed exception cases as the one presented in (Golding and Rosenbloom 1996). The indexing scheme is acquired by first converting symbolic rules

to neurules and then associating the produced neurules with the exception cases of the symbolic rules belonging to their merger sets.

The hybrid inference process combining neurule-based with case-based reasoning focuses on neurules (i.e. neurule-based reasoning). If an adequate number of the conditions of a neurule are fulfilled so that it can fire, firing of the neurule is suspended and CBR is performed for its indexed exception cases. CBR results are evaluated as in (Golding and Rosenbloom 1996) to assess whether the neurule will fire or whether the conclusion proposed by the exception case will be considered valid.

9 Conclusions

In this paper, we present an overview of our main research work involving neurules, a type of hybrid neuro-symbolic rules. An attractive feature of neurules is that compared to other connectionist approaches they retain the modularity and to some degree the naturalness of symbolic rules. In contrast to most neuro-symbolic approaches, a neurule-based system also provides an interactive inference mechanism and explanation facilities. We outlined aspects regarding construction of neurules from symbolic rule bases or empirical data, efficient updating of a neurule base constructed from symbolic rule bases or empirical data, neurule-based inference and combination of neurules with case-based reasoning.

Neurules have been used in developing an Intelligent Tutoring System (Prentzas, Hatzilygeroudis and Garofalakis 2002, Hatzilygeroudis and Prentzas 2004b). Intelligent Tutoring Systems (ITSs) require discrete knowledge bases to perform tasks of their different units (i.e. user modeling unit, pedagogical unit). Neurules facilitated the development and performance of the ITS since they satisfy most of the representation requirements concerning ITSs (Hatzilygeroudis and Prentzas 2004d, 2006). More specifically, neurule bases can be constructed from alternative knowledge sources producing a natural and modular representation scheme. Incremental development of neurule bases is also supported to accommodate source knowledge changes. Furthermore, neurule-based inference is natural, robust and time-efficient.

Our future work is directed to a number of aspects. Such aspects involve finding ways to (a) improve the neurule-based inference efficiency, (b) provide natural explanations, (c) incorporate fuzziness into neurules and (d) improve the mechanisms constructing neurules. Another future direction will involve use of neurules in different applications.

References

Aamodt A, Plaza E (1994) Case-based reasoning: foundational issues, methodological variations and system approaches. AI Communications 7:39–59.

Andrews R, Diederich J, Tickle A (1995) A survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-Based Systems 8:373-389

Bader S, Hitzler P (2005) Dimensions of neural-symbolic integration – a structured survey. In: Artemov S, Barringer H, d'Avila Garcez AS, Lamb LC, Woods J (eds) We Will Show Them: Essays in Honour of Dov Gabbay, volume 1, pages 167–194. International Federation for Computational Logic, College Publications

Bader S, Hitzler P, Holldobler (2008) Connectionist model generation: a first-order approach. Neurocomputing 71:2420-2432

Bookman L, Sun R (eds) (1993) Special issue on integrating neural and symbolic processes. Connection Science 5(3-4)

Browne A, Sun R (2001) Connectionist inference models. Neural Networks 14:1331-1355

Cloete I, Zurada JM (eds) (2000) Knowledge-based neurocomputing. The MIT Press, Cambridge

Frank A, Asuncion A (2010) UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. Accessed October 9, 2010

Fu LM (1993) Knowledge-based connectionism for revising domain theories. IEEE Transactions on Systems, Man, and Cybernetics 23:173-182

Fu LM (ed) (1994) Proceedings of the International Symposium on Integrating Knowledge and Neural Heuristics. Pensacola, Florida.

Gallant SI (1988) Connectionist expert systems. Communications of the ACM 31:152-169

Gallant SI (1993) Neural network learning and expert systems. The MIT Press, Cambridge

d'Avila Garcez AS, Broda K, Gabbay DM (2002) Neural-symbolic learning systems: foundations and applications, Perspectives in Neural Computing. Springer-Verlag

d'Avila Garcez A, Gabbay D, Holldobler S, Taylor J (2004) Special issue on neural-symbolic systems. Journal of Applied Logic 2

d'Avila Garcez A, Lamb LC, Gabbay DM (2007) Connectionist modal logic: representing modalities in neural networks. Theoretical Computer Science 371:34-53

Ghalwash AZ (1998) A recency inference engine for connectionist knowledge bases. Applied Intelligence 9:201-215

Golding AR, Rosenbloom PS (1996) Improving accuracy by combining rule-based and case-based reasoning. Artificial Intelligence 87:215-254

Gonzalez A, Dankel D (1993) The engineering of knowledge-based systems: theory and practice. Prentice-Hall, Upper Saddle River, NJ

Hatzilygeroudis I, Prentzas J (2000a) Neurules: integrating symbolic rules and neurocomputing. In: Fotiades D, Nikolopoulos S (Eds) Advances in Informatics. World Scientific Publishing

Hatzilygeroudis I, Prentzas J (2000b) Neurules: improving the performance of symbolic rules. International Journal on AI Tools 9:113-130

Hatzilygeroudis I, Prentzas J (2001a) An efficient hybrid rule based inference engine with explanation capability. In: Kolen J, Russell I (eds) Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference. AAAI Press, Menlo Park, CA

Hatzilygeroudis I, Prentzas J (2001b) Constructing modular hybrid knowledge bases for expert systems. International Journal on AI Tools 10:87-105

Hatzilygeroudis I, Prentzas J (2004a) Neuro-symbolic approaches for knowledge representation in expert systems. International Journal on Hybrid Intelligent Systems 1:111-126

Hatzilygeroudis I, Prentzas J (2004b) Using a hybrid rule-based approach in developing an intelligent tutoring system with knowledge acquisition and update capabilities. Expert Systems with Applications 26:477-492

Hatzilygeroudis I, Prentzas J (2004c) Integrating (rules, neural networks) and cases for knowledge representation and reasoning in expert systems. Expert Systems with Applications 27:63-75

Hatzilygeroudis I, Prentzas J (2004d) Knowledge representation requirements for intelligent tutoring systems. In: Lester JC, Vicari RM, Paraguacu F (eds) Proceedings of the Seventh International Conference on Intelligent Tutoring Systems, Lecture Notes in Computer Science, vol. 3220, Springer-Verlag

Hatzilygeroudis I, Prentzas J (2006) Knowledge representation in intelligent educational systems. In: Ma Z (ed) Web-based intelligent e-learning systems: technologies and applications. Idea Group Inc, Hershey, PA

Hatzilygeroudis I, Prentzas J (2010) Integrated rule-based learning and inference. IEEE Transactions on Knowledge and Data Engineering 22:1549-1562

Haykin S (2008) Neural Networks and Learning Machines. Prentice Hall, Upper Saddle River, NJ

Hilario M (1997) An overview of strategies for neurosymbolic integration. In: Sun R, Alexandre E (eds) Connectionist-symbolic integration: from unified to hybrid approaches. Lawrence Erlbaum Associates, Mahwah, NJ

Holldobler S, Kalinke Y (1994) Towards a massively parallel computational model for logic programming. In Proceedings of ECAI94 Workshop on Combining Symbolic and Connectionist Processing pp 68–77. ECCAI.

Komendantskaya E, Lane M, Seda AK (2007) Connectionist representation of multi-valued logic programs. In: Hammer B, Hitzler P (eds) Perspectives of Neural-Symbolic Integration. Springer-Verlag

McGarry K, Wermter S, MacIntyre J (1999) Hybrid neural systems: from simple coupling to fully integrated neural networks. Neural Computing Surveys 2:62-93

Medsker LR (1995) Hybrid intelligent systems. Kluwer Academic Publishers

Prentzas J, Hatzilygeroudis I (2002) Integrating hybrid rule-based with case-based reasoning. In: S. Craw, A. Preece (eds) Proceedings of the Sixth European Conference on Case-Based Reasoning - Advances in Case-Based Reasoning, Lecture Notes in Artificial Intelligence, vol. 2416. Springer-Verlag

Prentzas J, Hatzilygeroudis I, Garofalakis J (2002) A Web-based intelligent tutoring system using hybrid rules as its representational basis. In: Cerri SA, Gouarderes G, Paraguacu F (eds): Proceedings of the Sixth International Conference on Intelligent Tutoring Systems, Lecture Notes in Computer Science, vol. 2363. Springer-Verlag

Prentzas J, Hatzilygeroudis I (2005) Rule-based update methods for a hybrid rule base. Data and Knowledge Engineering 55:103-128

Prentzas J, Hatzilygeroudis I (2006) Construction of neurules from training examples: a thorough investigation. In: Garcez A, Hitzler P, Tamburini G (eds) Proceedings of the ECAI'2006 Workshop on Neural-Symbolic Learning and Reasoning

Prentzas J, Hatzilygeroudis I (2007a) Categorizing approaches combining rule-based and case-based reasoning. Expert Systems 24:97-122

Prentzas J, Hatzilygeroudis I (2007b) Incrementally updating a hybrid rule base based on empirical data. Expert Systems 24:212-231

Prentzas J, Hatzilygeroudis I (2009) Combinations of case-based reasoning with other intelligent methods. International Journal of Hybrid Intelligent Systems 6:189-209

Reichgelt H (1991) Knowledge representation, an AI perspective. Ablex, New York, NY

Souici-Meslati L, Sellami M (2006) Toward a generalization of neuro-symbolic recognition: an application to Arabic words. International Journal of Knowledge-based and Intelligent Engineering Systems 10:347-361

Sun R, Alexandre E (eds) (1997) Connectionist-symbolic integration: from unified to hybrid approaches. Lawrence Erlbaum Associates, Mahwah, NJ

Teng T-H, Tan Z-M, Tan A-H (2008) Self-organizing neural models integrating rules and reinforcement learning. In Proceedings of the IEEE International Joint Conference on Neural Networks. IEEE

Towell G, Shavlik J (1994) Knowledge-based artificial neural networks. Artificial Intelligence 70:119-165

Wermter S, Sun R (eds) (2000) Hybrid neural systems. Springer-Verlag

Xianyu JC, Juan ZC, Gao LJ (2008) Knowledge-based neural networks and its application in discrete choice analysis. In Proceedings of the Fourth International Conference on Networked Computing and Advanced Information Management. IEEE Computer Society

Yu L, Wang L, Yu J (2008) Identification of product definition patterns in mass customization using a learning-based hybrid approach. International Journal of Advanced Manufacturing Technologies 38:1061-1074