Published in the International Journal on Artificial Intelligence Tools (IJAIT),
Vol. 9, No. 1 (2000) 113-130.

NEURULES: IMPROVING THE PERFORMANCE OF SYMBOLIC RULES

I. HATZILYGEROUDIS, J. PRENTZAS
University of Patras, School of Engineering
Dept of Computer Engin. & Informatics, 26500 Patras, Hellas (Greece)
Email: ihatz/prentzas@ceid.upatras.gr
&
Computer Technology Institute, P.O. Box 1122, 26110 Patras, Hellas (Greece)

ABSTRACT

In this paper, we present a method for improving the performance of classical symbolic rules. This is achieved by introducing a type of hybrid rules, called neurules, which integrate neurocomputing into the symbolic framework of production rules. Neurules are produced by converting existing symbolic rules. Each neurule is considered as an adaline unit, where weights are considered as significance factors. Each significance factor represents the significance of the associated condition in drawing the conclusion. A rule is fired when the corresponding adaline output becomes active. This significantly reduces the size of the rule base and, due to a number of heuristics used in the inference process, increases efficiency of the inferences.

Keywords: hybrid knowledge representation, hybrid inference, symbolic representation, connectionist representation, production rules, neural networks.

1. Introduction

Most of existing expert systems are rule-based, that is the basis of their knowledge representation (KR) language is *symbolic rules*, often called if-then rules.[1] This is due to the very important benefits that production rules offer to knowledge representation and reasoning in expert systems, such as naturalness, modularity and ease of explanation. Rules are a representative of what is called *symbolic representation*.

Recently, popularity of using *artificial neural networks (ANNs)* in constructing expert systems has significantly increased.[2,3] A new category of expert systems, called connectionist expert systems, has been emerged.[4,5] ANNs provide a totally different approach to knowledge representation and reasoning from traditional (i.e. symbolic) AI. The main advantages of this approach are: representation of very complex and imprecise relationships, learning from experience and computational efficiency. ANNs are representatives of what is called *connectionist* or *subsymbolic representation*.

Nowadays, there has been extensive research activity at combining (or integrating) the symbolic and the connectionist approaches.[6,7,8] Two major categories of approaches to that integration have been distinguished, the unified and the hybrid.[9] The former is based on neural networks alone, whereas the latter on both the symbolic and the connectionist representations. There are a number of efforts at combining symbolic (classical or fuzzy) rules and neural networks for knowledge representation. Some of them follow the unified approach,[3,10,11,12] whereas others follow a semi-hybrid approach.[13,14,15,5] A weak point of both approaches is that the resulted systems lack the naturalness, modularity and explanation capabilities of symbolic rules.

The real hybrid systems include both, symbolic and connectionist structures and processes, and achieve an effective interaction among them. An interesting subcategory of real hybrids are those conforming to the subprocessing model, where typically the connectionist part is embedded into the symbolic part that act as the main problem solver.[9] Existing efforts to combine rule-based representations and ANNs that belong to this subcategory[16,17] use a loose coupling between the two components and serve different objectives. LAM[17], an expert system for window glazing design, uses ANNs as partial problem solvers to assist the main problem solver, a rule-based system. Setheo[16] is a logic-based system (a theorem prover) that uses the ANNs approach to improve the performance of its inference mechanism. An ANN is used to select the most promising node at each inference step in the search space.

In this paper, we mainly introduce a method for improving the performance of classical symbolic rules. This is achieved via *neurules*, a hybrid rule category also introduced here, that incorporate neurocomputing within the symbolic framework of production rules, in a way that increases their efficiency. Neurules are produced by converting existing symbolic rules. Our effort has a similar objective with that in Setheo: to use ANNs to improve performance of a symbolic representation. However, we achieve it by a uniform and tight integration of a symbolic component (production rules) and a connectionist one (the adaline unit) rather than a loose one. This paper is a revised and extended version of the one presented at ICTAI'99.[18]

The structure of the paper is as follows. Section 2 presents the hybrid formalism and corresponding system architecture. In Section 3, the algorithm for converting a symbolic rule-base into a hybrid one is described. Section 4 deals with the inference mechanism as well as various inference heuristics. An example illustrating the inference process of the integrated formalism is presented in Section 5 and experimental results in Section 6. Finally, Section 7 concludes.

2. The Hybrid Formalism

2.1. *Neurules*

We introduce *neurules* (: *neu*ral *rules*), a kind of hybrid rules[19]. Each neurule is considered as an adaline unit (Fig.1a). The *inputs* C_i (i = 1...n) of the unit are the *conditions* of the rule. Each condition C_i is assigned a number sf_i, called a *significance factor*, corresponding to the weight of the corresponding input of the adaline unit. Moreover, each rule itself is assigned a number sf_0, called the *bias factor*, corresponding to the weight of the *bias input* of the unit (C_0= 1, not illustrated in Fig.1 for the sake of simplicity).

Each input takes a value from the following set of discrete values:

$$C_i = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{if condition is false} \\ 0.5 & \text{if value is unknown} \end{cases} \quad (1)$$

This gives the opportunity to easily distinguish between the falsity (C_i = 0) and the absence (C_i = 0.5) of a condition, in contrast to symbolic rules. The choice of this set of values is strongly related to the inference heuristics, introduced in Section 4.2. This means that if they change, the heuristics may not be still valid.

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

$$D = f(\mathbf{a}), \quad \mathbf{a} = sf_0 + \sum_{i=1}^{n} sf_i C_i \quad (2)$$

as usual [2,3], where **a** is the *activation value* and *f(x)* the *activation function*, which is a threshold function (Fig.1b). Hence, the output can take one of two values, '-1' and '1', representing 'failure' and 'success' of the rule respectively.

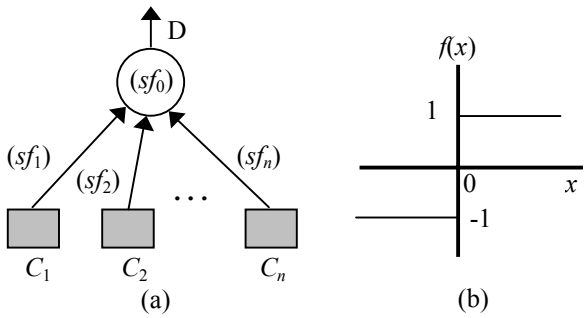


Fig. 1. (a) a neurule as an adaline unit (b) the activation function

2.2. *Syntax and semantics*

The general syntax (structure) of a rule is:

<rule> ::= [(<bias-factor>)] **if** <conditions> **then** <conclusions>
<conditions> ::= <condition> {**,** <condition>}
<conclusions> ::= <conclusion> {**,** <conclusion>}
<condition> ::= <variable> <l-predicate> <value-object>
[(<significance-factor>)]
<conclusion> ::= <variable> <r-predicate> <value-object>.

By <variable> a *variable* is denoted, that is a symbol representing a concept in the domain, e.g. "sex", "pain" in a medical domain. We distinguish three types of variables:

- *askable variables,* that is variables for which the user will be prompted to give a value.
- *goal variables,* that is variables constituting the goals of the inference process.
- *inferable variables,* that is variables constituting intermediate goals of the inference process.

All variables are single-valued, that is they can take only one value at a time, and their values are disjoint, that is the one excludes the others.

By <l-predicate> a symbolic or a numeric predicate is denoted. The *symbolic predicates* are {is, isnot}, whereas the *numeric predicates* are {<, >, =}. Conditions including the 'is' (resp. 'isnot') predicate are called *is-conditions* (resp. *isnot-conditions*). <r-predicate> can only be a symbolic predicate. <value-object> denotes a value. It can be a *symbol* or a *number*. Finally, <bias-factor> and <significance-factor> are real numbers. The significance factor of a condition represents the significance (weight) of the condition in drawing the conclusion(s). So, the semantics of significance factors is quite different from that of certainty factors or probabilities.

R_1:
if sex **is** man **,**
age **>** 20 **,**
age **<** 36
then patient_class **is** man21_35

(a)

N_1:
(-8) **if** pain **is** continuous (5) **,**
patient_class **isnot** man36_55 (2.5) **,**
fever **is** medium (2) **,**
fever **is** high (2)
then disease_type **is** inflammation

(b)

Fig. 2. An example (a) symbolic rule and (b) neurule

The significance factors and the bias factor are optional in a rule (which is denoted by '[ ]'). Thus, neurules (with factors) and symbolic rules (without factors)

are equally supported by the representation formalism. (The terminal symbol "**,**" in the case of a symbolic rule denotes a conjunction). Two example rules, a symbolic and a neurule, from a medical diagnosis domain, are presented in Fig.2.

We distinguish between two types of neurules, the negative-bias and positive-bias rules. A *negative-bias rule* has a negative bias factor ($sf_0 < 0$), whereas a *positive-bias rule* a positive bias factor ($sf_0 > 0$).

2.3. *The hybrid architecture*

In Fig.3, the functional architecture of the hybrid rule-based system to implement the method for improving the performance of symbolic rules is presented. The run-time system (in the dashed rectangle) consists of three modules, functionally similar to those of a conventional rule-based system: the *hybrid rule base (HRB)*, the *hybrid inference mechanism (HIM)* and the *working memory (WM)*.

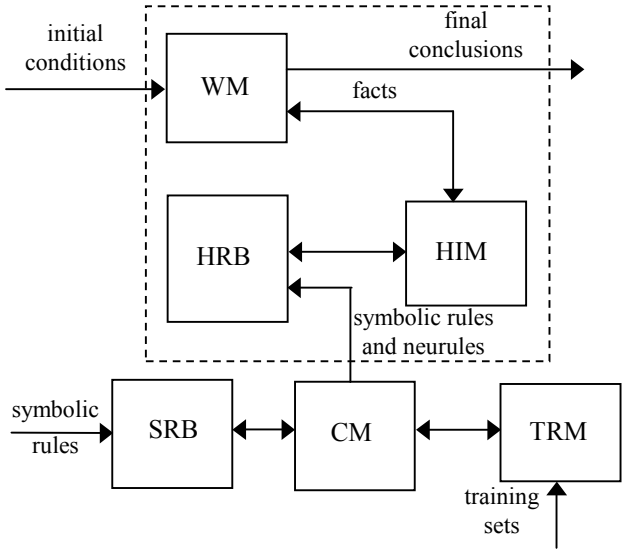


Fig. 3. The hybrid rule-based architecture

The HRB contains neurules and symbolic rules, produced by conversion from the *symbolic rule base (SRB)* via the *conversion mechanism (CM)*. The SRB contains only symbolic rules. The produced neurules are trained via the *training and reorganizing mechanism (TRM)*. The functionality of the CM and the TRM are explained in Section 3. So, production of the neurules takes place before run-time.

HIM is responsible for making inferences by taking into account the initial conditions in the WM and the rules in the HRB. HIM is discussed in Section 4.

WM contains *facts*. A fact has the same format as a condition/conclusion of a rule. However, it can additionally have as value the special symbol 'unknown'. Facts represent either initial conditions or intermediate/final conclusions produced during the course of an inference.

3. Converting Symbolic Rules to Neurules

As mentioned, the HRB is constructed by converting rules in the SRB to neurules. Conversion of a symbolic rule base into a hybrid rule base is achieved via the *symbolic-to-hybrid conversion algorithm*, outlined below:

(1) Construct merger sets from the SRB and form mergers. Put the non-merging rules in the HRB.
(2) For each merger, specify its training set, by selecting rows from the combined truth table of its merging rules.
(3) Train each merger and produce one or more neurules and possibly one or more remaining (symbolic) rules. Put the remaining rules in the HRB.
(4) Reorganize the conditions of the produced neurules and join them with the non-merging and remaining rules into the HRB.

In the sequel, we elaborate on each step of the algorithm.

3.1 *Merging symbolic rules*

A knowledge base (SRB) is initially constructed using only symbolic rules, either by interviewing experts or from domain data using machine learning techniques (like e.g. the ID3 algorithm). Then, the SRB is converted into the HRB via the CM.

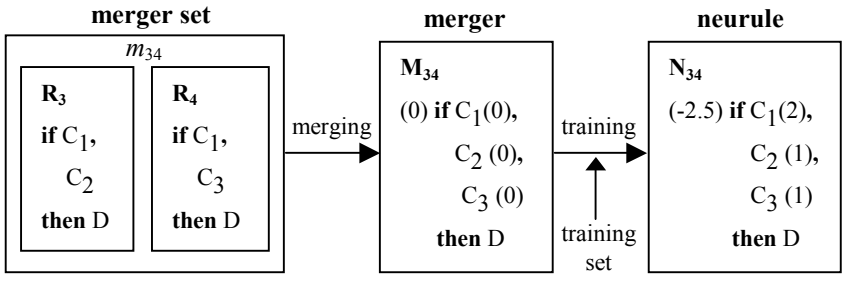


Fig. 4. Transforming a merger set into a neurule

Symbolic rules with the same conclusion, called *merging rules*, are selected into a *merger set*. Thus, a number of merger sets are produced from the SRB. Symbolic rules with a unique conclusion do not participate in a merger set, they constitute the *non-merging rules*. Then, rules in each merger set form a *merger*, which is a rule

having as conditions the conditions of all the rules in the corresponding merger set. A simple example is illustrated in Fig.4, where the merging rules R_3 and R_4, elements of the merger set $m_{34} = \{R_3, R_4\}$, are merged into merger M_{34} (zeros are initial values assigned to the factors).

3.2. *Training sets specification*

Before training, the training set of each merger is determined by selecting rows from the *combined truth table* of the rules in the corresponding merger set. The combined truth table is produced from the *combined logical function* of the rules, which is the disjunction of the conjunctions of the conditions of them. For example, the combined logical function of the rules in the merger set of Fig.4 is ((C_1 AND C_2) OR (C_1 AND C_3)) ≡ (C_1 AND (C_2 OR C_3)). The *training set* of a merger consists of training patterns, which are lists of the form [v_1 v_2 … v_n d], where v_i, i= 1,…,n are the condition values and d the (desired) output value. For example, the training set of M_{34} in Fig. 4 was T= ([1 0 0 -1], [1 1 0 1], [0 1 1 -1], [1 0 1 1])

Due to domain specific reasons, not all of the rows of a combined truth table are valid for training the corresponding merger. So, a selection should be made. Valid training set selection is the guarantee that the produced HRB is correct and equivalent to the SRB. Therefore, we devised a number of criteria to remove invalid rows. First, we introduce the notion of *related conditions*:

- Two conditions are *related* if they refer to the same variable.

For example, “fever is low” and “fever is high” are related conditions, because they refer to the same variable ‘fever’. Now, we introduce the following *invalid-row criteria*:

- Related is-conditions (resp. isnot-conditions) cannot be simultaneously true (resp. false).
- Related is-conditions (resp. isnot-condidtions) with exhaustive values cannot be simultaneously false (resp. true).
- An is-condition and an isnot-condition that are related and have the same value (e.g. “fever is high”, “fever isnot high”) cannot be simultaneously true.

We further introduce a non-literally detected invalid-row criterion, that is one that needs expert’s help:

- Two conditions are *inconsistent* if they cannot really happen to be simultaneously true, due to pragmatic reasons.

Rows that do not meet the above criteria, should be removed from the truth table, and not used in the training set.

For example, consider the rules “**if** C_1, C_2 **then** D”, and “**if** C_3, C_4 **then** D”, which constitute a merger set, and their combined logical function, ((C_1 AND C_2) OR (C_3 AND C_4)). If C_1 ≡ “fever is high” and C_3 ≡ “fever is low”, then they are related is-conditions, as they refer to the same variable ‘fever’. Thus, four rows in the truth table of the combined function that have ‘1’ in the places of both C_1 and C_3

should be removed. If C_2, C_4 are also related conditions, three more rows are removed. If C_1 is inconsistent with C_4, one more row is removed.

So, the remaining rows, after application of the invalid-row criteria to the combined truth table of a merger, constitute the training set of the merger. It is clear from the above example, that the number of the training patterns in a training set can be significantly reduced compared to those in the truth table. So, application of the invalid-row criteria makes training, apart from valid, less complex and less time consuming.

3.3. *Training mergers*

After the above has been done, each merger is individually trained via the TRM, using its training set, to produce one or more neurules (see Fig. 4) and possibly one or more symbolic rules. The standard least mean square (LMS) learning algorithm is employed to calculate the values of the factors to be assigned to each neurule.

However, there are cases where the LMS algorithm fails to specify the right significance factors for a number of neurules. That is, the corresponding adaline units of those rules do not correctly classify some of the training patterns. This means that the patterns in the training set correspond to a *non-separable (boolean) function*.It is known that the adaline model cannot fully represent such functions.[2,3]

To overcome this problem, we successively split the corresponding merger sets into subsets until the right factors are determined. Splitting is made in a way that the produced subsets contain *close rules*, that is rules with as more common or related conditions as possible. So, two or more neurules may be produced. In such a situation, a merger subset may contain just one symbolic rule. This is a *remaining rule*. Thus, an initial merger set with a non-separable training set will produce more than one neurule and possibly one or more symbolic rules (remaining rules). Hence,step 3 in the symbolic-to-hybrid conversion algorithm is analyzed as follows:

3.1 Train each merger using the corresponding training set.

3.2 If training fails split its merger set in two subsets of comparable size, such that the rules in each subset have as more common or related conditions as possible.

3.3 For each merger subset, apply step 3.1 recursively until either training succeeds or the merger subset becomes a singleton (remaining rule).

3.4. *Reorganizing neurules*

After training, the conditions of a neurule are distributed between two groups, the negative group and the positive group. The *negative group* includes the conditions with negative significance factors, whereas the *positive group* those with positive factors. The conditions in the positive group of a negative-bias rule are ranked in *descending order*, according to the values of their significance factors. Furthermore, the conditions in its negative group are put in front of those in its positive one.

Similarly, the conditions in the negative group of a positive-bias rule are ranked in *ascending order*. Also, the conditions in its positive group are put in front of those in its negative group. Reorganization of the neurules is done to support the inference heuristics introduced in Section 4.2.

4. The Hybrid Inference Mechanism

4.1. *Basic terminology*

The *hybrid inference mechanism* (HIM) is based on a backward chaining strategy. There are two stacks used, a *goal stack (GS)*, where the *current goal* (condition) G_c to be matched is always on its top, and a *rule stack (RS)*, where the *current rule* R_c under evaluation is always on its top. The conflict resolution strategy, due to backward chaining and the neurules, is based on textual order. A rule succeeds if it *evaluates* to 'true', that is all of its conditions evaluate to 'true', in the case of a symbolic rule, or its output is computed to be '1' after evaluation of its conditions, in the case of a neurule.

A condition evaluates to 'true', if it matches a fact in the WM, 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 WM with the same variable. In this case, either a question is made to the user to provide a value for the variable, in the case of an askable variable, or a rule in the HRB with a conclusion containing that variable is examined, in case of an inferable variable. A condition evaluates to 'false' if there is a fact in the WM with the same variable, predicate and different value, in case of an askable variable, and additionally there is no rule in the HRB that has a conclusion with the same variable, in case of an inferable variable. Inference stops either when a rule with a goal variable is fired (success) or there is no further action (failure).

4.2. *Inference heuristics*

To increase inference efficiency, a number of heuristics are used. Most of them are based on the selected condition values (see formula (1), Section 2.1) and the activation function (see Fig.1b). We should mention that the activation value is *incrementally computed*, that is contribution of each condition to the weighted sum in formula (2) (Section 2.1) is added immediately after its evaluation.

4.2.1. *Ordered condition evaluation*

There are different strategies followed for the evaluation of the negative-bias and the positive-bias rules. When computing the activation value of a negative-bias rule, first the conditions in the negative group are evaluated and then those in the positive group. Thus, during evaluation of the conditions in the positive group, as soon as the

result exceeds the threshold (0), evaluation stops and the output gets the value '1' ('true'). This is so, because it cannot change in any case, if proceed. Also, since the conditions in the positive group of a negative-bias rule are ranked in a descending order, the positively 'heavier' conditions are first evaluated, then the 'lighter', thus speeding up the computation.

When computing the activation value of a positive-bias rule, first the factors in the positive group are evaluated and then the factors in the negative group. Thus, during evaluation of the conditions in the negative group, as soon as the result becomes less than the threshold (0), evaluation stops and the output gets the value '-1' (false). Again, this is so, because it cannot change in any case, if proceed. Also, since the conditions in the negative group of a positive-bias rule are ranked in ascending order, the negatively 'heavier' conditions are first evaluated, then the 'lighter', thus speeding up the computation.

4.2.2. *Remaining sum criterion*

The remaining sum criterion is a generalization of the 'critical condition situation' heuristic.[20,19]

N_5

(-8.5) **if** C_1(4),
C_2 (3),
C_3 (2),
C_4 (2),
C_5 (2),
C_6 (1)
then D

C_i (true/false)	**a**	**sum** (a-sf_i)
-	-8.5	14
C_1 (false)	-8.5	10
C_2 (true)	-5.5	7
C_3 (false)	-5.5	5
|-5.5| > 5:	evaluation stops (failure)	

Fig. 5. Remaining sum criterion example

In the case of a negative-bias rule, as soon as a condition in the positive group has been evaluated, the weighted sum (**a**) is compared to the *remaining sum* (**sum**), that is the sum of the factors of the conditions in the group that have not been evaluated yet. If |**a**| > **sum**, then it is certain that **a** will remain negative and the rule will evaluate to '-1' ('false'), even if all remaining conditions evaluate to 'true'. So, the remaining conditions in the positive group do not have to be evaluated and evaluation stops.

Similarly, in the case of a positive-bias rule, the weighted sum is compared to the remaining sum of the factors of the conditions in the negative group that have not been evaluated. If **a** > |**sum**|, then it is certain that **a** will remain positive and the rule will evaluate to '1' ('true'), even if all remaining conditions fail. So, the remaining

conditions in the negative group do not have to be evaluated and evaluation stops. The table in Fig.5 shows a tracing of the evaluation of the conditions of neurule N_5, based on this heuristic.

4.2.3. *Related conditions situation*

The last heuristic concerns related conditions in a neurule. If one of the related is-conditions of a neurule evaluates to 'true', the others fail, since their values are disjoint. For example, in neurule N_1 in Fig.2, if "fever is medium" evaluates to 'true', there is no need to evaluate the condition "fever is high", because it certainly fails. Also, if one of them evaluates to 'unknown', the rest evaluate to 'unknown' too.

4.3. *The inference process*

In this section, we present the inference process of the integrated formalism, performed by the HIM, via the definitions of the procedures: MAIN, STACK-SET, RULE-SEARCH, RULE-EVAL, SYMB-EVAL and NEUR-EVAL below. In the following, $\mathbf{a}_{R_C}$ represents the activation value (i.e. the weighted sum) of the current neurule, and C_{R_C} represents the condition of the current neurule where evaluation is suspended, to search for a rule to evaluate the condition. For the sake of simplicity, rules with only one conclusion are supposed and the related condition situation heuristic is not supported.

MAIN (G_0)
1. Put the initial goal (G_0) on *GS*
2. While RULE-SEARCH returns 'true'
   2.1 Call STACK-SET
3. Stop (failure).

STACK-SET
1. While RULE-EVAL(R_C) returns 'true'
   1.1 Remove R_C from *RS*
   1.2 Move G_C to *WM*
   1.3 If *GS* is empty
      1.3.1 Stop (success).
2. Remove R_C from *RS*

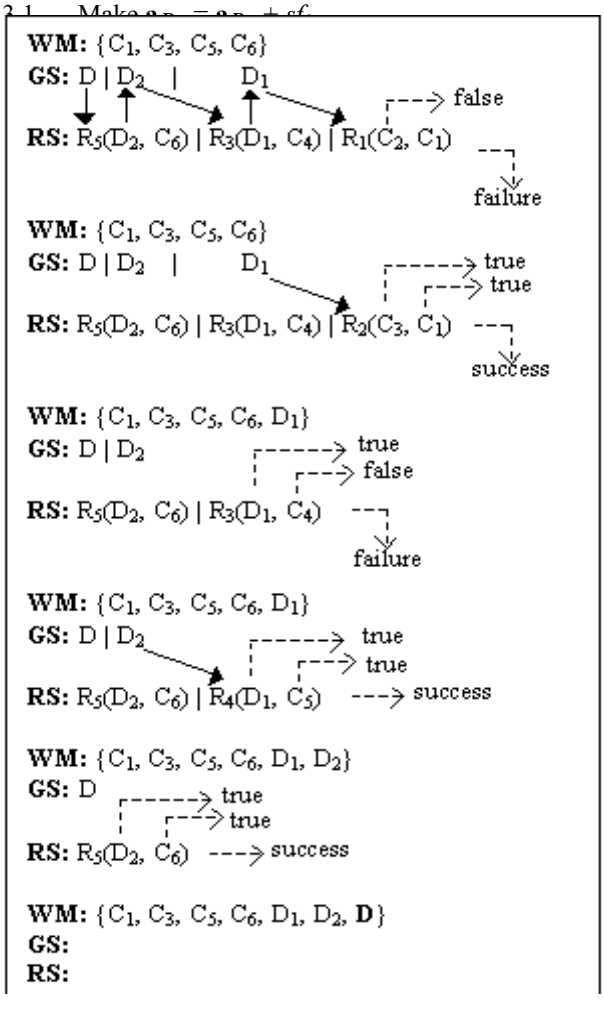
RULE-SEARCH
1. For each R_i in the *HRB*
   1.1 If R_i matches G_C
      1.1.1 Set $R_C = R_i$; $C_{R_C} = \varnothing$
      1.1.2 Return 'true'
   1.2 Return 'false'.

RULE-EVAL(R_C)

1. If R_c is a symbolic rule
   1.1 Call SYMB-EVAL.
2. Call NEUR-EVAL

SYMB-EVAL(R_c)
1. For each C_i of R_c
   1.1 If C_i cannot be evaluated
      1.1.1 Make $G_c = C_i$
      1.1.2 Call RULE-SEARCH
         1.1.2.1 If it returns 'false', stop (failure).
2. Return 'true'.

NEUR-EVAL(R_c)
1. If $C_{R_c} = \varnothing$
   1.1. set $\mathbf{a}_{R_c} = sf_0$.
   1.2. If $sf_0 > 0$, set sum = $\sum_{sf_i<0} sf_i$
   1.3. If $sf_0 < 0$, set sum = $\sum_{sf_i>0} sf_i$
2. For each C_i of R_c after C_{R_c}
   2.1 If $sf_0 * sf_i < 0$, set sum = sum + ($-sf_i$)
   2.2 If C_i cannot be evaluated,
      2.2.1. Set $G_c = C_i$; $C_{R_c} = C_i$
      2.2.2. Call RULE-SEARCH
         2.2.2.1. If it returns 'false', go to 2.5
         2.2.2.2. Call STACK-SET
   2.3 If C_i evaluates to 'true'
      2.3.1. Make $\mathbf{a}_{R_c} = \mathbf{a}_{R_c} + sf_i$
2
2
2
2
3. F



5. An E

We pres ned in our hybrid f

Let consider the following SRB.

R_1: **if** C_2 , C_1 **then** D_1
R_2: **if** C_3 , C_1 **then** D_1
R_3: **if** D_1 , C_4 **then** D_2
R_4: **if** D_1 , C_5 **then** D_2
R_5: **if** D_2 , C_6 **then** D

Suppose that WM = {C_1, C_3, C_5, C_6} and our (initial) goal is D. The symbolic inference st[illegible] solid arrow means "puts [illegible]".

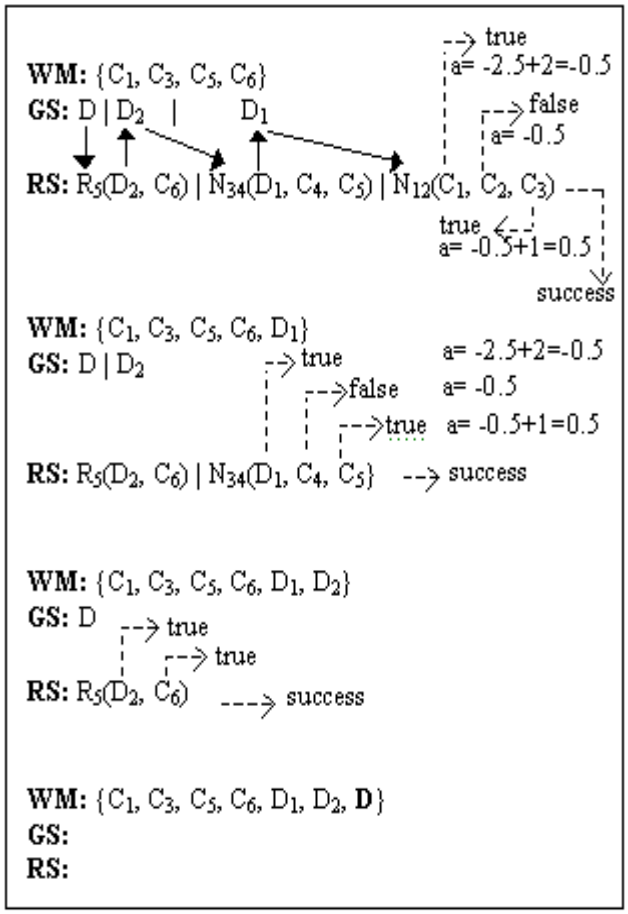


Fig. 6b. Hybrid inference

The HRB, after merging {R_1, R_2} and {R_3, R_4}, is:

N_{12}: (-2.5) **if** C_1 (2) , C_2 (1) , C_3 (1) **then** D_1
N_{34}: (-2.5) **if** D_1 (2) , C_4 (1) , C_5 (1) **then** D_2

R_5: **if** D_2 , C_6 **then** D

The hybrid inference steps to prove D are illustrated in Fig.6b (top-down). It is clear that symbolic inference is more expensive than the hybrid one. It should be also noticed that hybrid inference is much less dependent on the order of the rules than symbolic inference. For example, if R_1 and R_2 as well as R_3 and R_4 were interchanged the symbolic inference would be shortened, whereas interchange of N_{12} and N_{34} would make no difference for the hybrid inference.

6. Experimental Results

The symbolic-to-hybrid conversion algorithm was applied to two medical rule bases consisting of fifty-nine (59) and one-hundred-thirty-four (134) symbolic rules respectively. These initial symbolic rule bases will be referred to as SRB1 and SRB2 respectively, whereas the resulting hybrid rule bases as HRB1 and HRB2.

Eighteen (18) of the symbolic rules in SRB1 had a unique conclusion and thus couldn't be merged (non-merging rules). The rest forty-one (41) rules formed ten merger sets. From those, four had to split. Two of them were split in two and the other two in three subsets. From the so formed ten subsets, four were singletons (remaining rules). Finally, the total number of rules in the HRB1 was thirty-four (34), twenty-two (22) symbolic rules and twelve (12) neurules. So, we had an over 40% reduction in the number of the rules. All of the neurules were negative-bias rules. It should be noted that, due to the invalid-row criteria, over 38%, in average, of the rows of the combined truth tables of the merging sets were removed.

Table 2. SRB1 vs. HRB1 inferences

Inference No	Rules Visited	Conditions Evaluated	Decision
	Symb. / Hybr.	Symb. / Hybr.	
1	2 / 2	6 / 6	Inflammation
2	5 / 3	12 / 11	Inflammation
3	9 / 4	15 / 12	Inflammation
4	25 / 12	35 / 31	Arthritis
5	22 / 9	41 / 27	Prim. Malignant
6	22 / 12	46 / 35	Prim. Malignant
7	30 / 13	46 / 33	Prim. Malignant
8	41 / 18	50 / 35	Dec. Metabolical
9	53 / 27	67 / 64	Second. Malignant
Total	**209 / 100**	**318 / 254**	

Thirty-seven (37) of the symbolic rules in the SRB2 had a unique conclusion and thus couldn't be merged (non-merging rules). The rest ninety-seven (97) rules formed twenty merger sets. From those, thirteen had to be split. From the resulting subsets thirty were singletons (remaining rules). The splitting degree of the merger

sets was higher here than SRB1, due to the fact that their rules had fewer common conditions. Finally, the total number of rules in the HRB2 was ninety (90), sixty-seven (67) symbolic and twenty-three (23) neurules. So, we had an over 30% reduction in the number of rules, which is less than the one in HRB1, because of the higher splitting degree. Once again, all of the neurules were negative-bias rules. Due to the invalid-row criteria, over 39%, in average, of the rows of the combined truth tables were removed.

Table 3. SRB2 vs. HRB2 inferences

Inference No	**Rules Visited**	**Conditions Evaluated**	**Decision**
	Symb. / Hybr.	**Symb. / Hybr.**	
1	5 / 3	15 / 12	Early-inflammation
2	9 / 5	19 / 13	Early-inflammation
3	13 / 6	38 / 16	Inflammation-or-benign-tumor
4	29 / 10	61 / 36	Soft-tissue-inflammation
5	34 / 15	39 / 26	Soft-tissue-inflammation
6	38 / 17	48 / 32	Soft-tissue-inflammation
7	45 / 21	106 / 54	Very-early-bone-inflammation
8	50 / 22	122 / 62	Soft-tissue-early-bone-inflammation
9	55 / 26	67 / 44	Soft-tissue-bone-inflammation
10	62 / 30	82 / 54	Early-soft-tissue-inflammation
11	76 / 41	157 / 92	Bone-inflammation-or-benign-tumor
12	89 / 51	183 / 125	Intensive-soft-tissue-early-bone-inflammation
13	94 / 56	177 / 117	Intensive-soft-tissue-with-early-bone-inflammation
14	100 / 60	195 / 132	Osteomelitis-or-benign-tumor
15	106 / 64	144 / 100	Early-soft-tissue-and-bone-inflammation
16	110 / 68	166 / 117	Tumor-or-bone-inflammation
17	114 / 71	171 / 126	Bone-tumor-or-very-intensive-bone-inflammation
18	118 / 75	149 / 108	Retracted-or-early-bone-inflammation-or-benign-tumor
19	122 / 79	166 / 128	Malignant-or-vessel-enriched-benign-tumor
20	126 / 82	162 / 122	vessel-enriched-malignant-tumor-osteosarcoma
21	130 / 86	134 / 99	Benign-bone-neoplasia
22	134 / 90	154 / 120	Normal-radionucleus-study
Total	**1659 / 978**	**2555 / 1735**	

Inferences were proved to be equivalent for both cases in both bases, that is we had the same conclusions for the same variable-value data both in the symbolic and the hybrid cases in both bases. Equivalence is basically guaranteed by the fact that the neurules are trained with valid data from the combined truth table of the merging rules. However, inferences from SRB1 and SRB2 were proved to be longer, in terms of the rules visited, and more expensive, in terms of the conditions evaluated (which is a stronger evidence), than the corresponding ones from the HRB1 and HRB2. In Tables 2 and 3, a number of experimental results supporting this fact are presented.

As it is clear from the tables, in general, the number of visited rules in the hybrid case is much less than the one in the symbolic case. There is an average reduction of over 50% in the rules visited in the hybrid case in Table 2. This is not however directly reflected to the evaluated conditions. In average, there is only a reduction of about 20% in the evaluated conditions. This is mainly due to the fact that in the symbolic case failure of just one condition rejects the rule from further consideration, whereas in the hybrid case usually more than one condition (at least those as far as remaining sum criterion is fulfilled) should fail. Similarly, there is an over 40% average reduction in the rules visited and an over 30% in the conditions evaluated in the hybrid case in Table 3. There is a better reflection here, because of the larger number of rules. Also, in general, the longer an inference is the better the result is in the hybrid case.

7. Conclusions

In this paper, we mainly introduce a method for improving the performance of classical symbolic rules. This is achieved via neurules, a kind of hybrid rules, also introduced here.

Neurules integrate production rules and the adaline neural unit and are produced by converting symbolic rules from a symbolic knowledge base. In this way, the number of the rules in the knowledge base is drastically reduced, since each neurule is actually a merger of more than one symbolic rule.

On the other hand, inferences are more efficient for two reasons. First, the number of participating rules has been drastically reduced. Second, the number of the evaluated conditions has been also significantly reduced, due to embedded heuristics.

The adaline unit and the LMS algorithm are not of the most powerful existing mechanisms. The adaline unit cannot represent non-separable training patterns, so symbolic rules with the same conclusion may not be transformed into a single neurule. This is a weak point of the formalism as well as a good reason for further investigation. One could think of more complex neurules, e.g. made of a two layer neural net.

On the other hand, neurules retain in a large degree the benefits of symbolic rules, such as naturalness and modularity. Indeed, neurules are understandable, since each significance factor in a neurule represent the contribution of the corresponding

condition in drawing the conclusion. Also, one could add new or remove old neurules without much bothering about making any technical changes to the knowledge base. So, neurules could be considered as a new knowledge representation formalism. What is missing is a direct way of producing them from empirical data. This is one of our current research directions.[21]

References

[1] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems*, Eddison-Wesley, Reading, MA (1984).

[2] R. Hect-Nielsen, *Neurocomputing*, Addison-Wesley, Reading, MA (1990).

[3] S. I. Gallant, *Neural Network Learning and Expert Systems*, MIT Press (1993).

[4] S. I. Gallant, *Connectionist Expert Systems*, CACM, **31** (1988) 152-169.

[5] B. Boutsinas and M. N. Vrahatis, *Nonmonotonic Connectionist Expert Systems*, Proc. 2nd WSES/IEEE/IMACS International Conference on Circuits, Systems and Computers, Athens, Hellas (Oct. 1998).

[6] R. Sun and L. Bookman (Eds), *The Working Notes of the AAAI Workshop on Integrating Neural and Symbolic Processes: The Cognitive Dimension*, San Jose, California (July 1992).

[7] L. M. Fu (Ed), *Proceedings of the International Symposium on Integrating Knowledge and Neural Heuristics (ISIKNH'94)*, Pensacola, FL (May 1994).

[8] R. Sun and E. Alexandre (Eds), *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches*, Lawrence Erlbaum (1997).

[9] M. Hilario, *An Overview of Strategies for Neurosymbolic Integration*, ch.2 in [8].

[10] R. Sun, *Integrating Rules and Connectionism for Robust Commonsense Reasoning*, Sixth-Generation Computer Technology, John Wiley & Sons (1994).

[11] R. R. Yager, *Modelling and formulating fuzzy knowledge bases using neural networks*, *Neural Networks* **7(8)** (1994) 1273-1283.

[12] A. Z. Ghalwash, *A Recency Inference Engine for Connectionist Knowledge Bases*, Applied Intelligence **9** (1998) 201-215.

[13] L-M Fu and L-C Fu, *Mapping rule-based systems into neural architecture*, Knowledge-Based Systems **3** (1990) 48-56.

[14] F. Kozato and Ph. De Wilde, *How Neural Networks Help Rule-Based Problem Solving*, Proceedings of the ICANN'91 (1991) 465-470.

[15] J. M. Keller and H. Tahani, *Implementation of conjuctive and disjunctive fuzzy logic rules with neural networks*, International Journal of Approximate Reasoning **6(2)** (1992) 221-240.

[16] R. Letz, S. Bayerl and W. Bibel, *Setheo: A high-performance theorem prover*, Journal of Automated Reasoning **8(2)** (1992) 183-212.

[17] L.R. Medsker, *Hybrid Neural Networks and Expert Systems*, Kluwer Academic Publishers, Boston (1994).

[18] I. Hatzilygeroudis and J. Prentzas, *Neurules: Improving the Performance of Symbolic Rules*, Proceeedings of 11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'99), Chicago, IL (Nov. 1999) 417-424.

[19] I. Hatzilygeroudis, J. Prentzas, *Neurules: Integrating Symbolic Rules and Neurocomputing*, Proceedings of the 7th Hellenic Conference on Informatics, Ioannina, Hellas (Aug. 1999) V52-V60.

[20] I. Hatzilygeroudis, *Integrating rules and neurocomputing for knowledge representation*, in [7] (1994) 40-46.

[21] I. Hatzilygeroudis and J. Prentzas, *Producing Modular Hybrid Rule Bases for Expert Systems*, Proceedings of the 13th International FLAIRS Conference, Orlando, FL (May 2000) (forthcoming).