Published in in D. Fotiades and S. Nikolopoulos (Eds), “Advances in Informatics”, World Scientific Pub., 2000, 122-133.

NEURULES: INTEGRATING SYMBOLIC RULES AND NEUROCOMPUTING

I. HATZILYGEROUDIS, J. PRENTZAS

University of Patras, School of Engineering
Dept of Computer Engin. & Informatics, 26500 Patras, Hellas
&
Computer Technology Institute, P.O. Box 1122, 26110 Patras, Hellas

In this paper, a hybrid knowledge representation formalism that integrates neurocomputing into the symbolic framework of production rules is presented. This is achieved by introducing *neurules*, a type of integrated 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. In this way, naturalness and modularity of production rules are retained, and imprecise relations between the conditions and the conclusion of a rule can be represented. Additionally, a number of heuristics used in the inference procedure result in increasing efficiency.

1 INTRODUCTION

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

Recently, popularity of using what is called *connectionism* or *neurocomputing* in constructing expert systems has been significantly increased. A new category of expert systems, called connectionist expert systems [3], has emerged. Their basis is *artificial neural networks (ANNs)* that provide a totally different approach to knowledge representation and reasoning from traditional AI. The main advantages of this approach are its capabilities of representing very complex and imprecise relationships and learning from experience.

Nowadays, there has been extensive research activity at combining/integrating the symbolic and the neurocomputing approaches (see e.g. [6, 9]). To that end, there are a number of efforts at combining production rules and neural networks for knowledge representation [5]. Some of them follow the unified approach [3, 4, 8], whereas others follow a pseudo-hybrid approach [1, 2, 7], called the translational approach in [5]. A weak point of both approaches is that the resulted system lacks the naturalness and modularity of symbolic rules.

In this paper, we introduce a KR formalism which attempts to incorporate aspects of neurocomputing within the symbolic framework of production rules in a way that preserves their naturalness and modularity on the one hand, and increases their efficiency on the other.

The structure of the paper is as follows. Section 2 presents the integrated formalism. In Section 3, methods and mechanisms for constructing a knowledge base are described. Section 4 deals with the inference mechanism. An example is presented in Section 5 and some experimental results in Section 6. Finally, Section 7 concludes.

2 THE HYBRID FORMALISM

2.1 Integration Model

We introduce *neurules* (: *neu*ral *rules*) alongside *symbolic rules*. Each neurule is considered as an adaline unit (Figure 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. The bias factor adds flexibility to the model as far as computation of the factors is concerned.

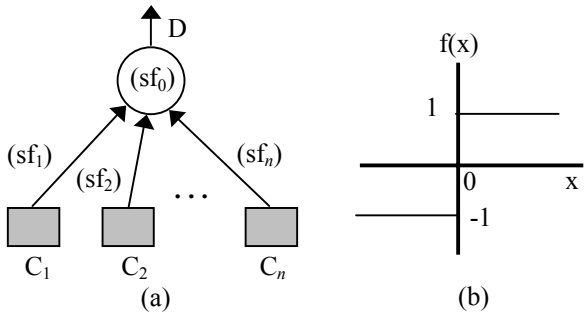

Figure 1. (a) a neurule as an adaline unit (b) the activation function

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

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

This gives the opportunity to easily distinguish between the falsity and the absence of a condition, in contrast to symbolic rules. The *output D*, which represents the *conclusion* (decision) of the rule, is calculated via the formulas:

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

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

2.2 Syntax and Semantics

The general syntax (structure) of a rule in our formalism is given below[1]:
<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>.
where <variable> denotes a *variable*, that is a symbol representing a concept in the domain, e.g. "sex", "pain" etc. <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-object> denotes a value. It can be a *symbol* or a *number*. Different types of <value-object> are associated with different predicates. Finally, <bias-factor> and <significance-factor> are real numbers.

R_1:
if sex **is** man**,**
age > 20**,**
age < 36
then patient_class **is** man_21_35

R_2:
(-8) **if** pain is continuous (5)**,**
patient_class **isnot** man_36_55 (2.5)**,**
fever **is** medium (2)**,**
fever **is** high (2)
then disease_type **is** inflammation.

Figure 2. A symbolic rule and a neurule.

As it is clear, significance factors and the bias factor are optional in a rule. Thus, neurules (with factors) and symbolic rules (without factors) are equally supported by our representation formalism. (The terminal symbol "**,**" in the case of a

[1] A BNF notation is used hereafter, where '[]' denotes optional occurrence and '{}' zero, one or more occurrences of the enclosed expression.

symbolic rule denotes a conjunction). Two example rules, a symbolic and a neurule, from a medical domain, are presented in Figure 2.

Apart from rules, our formalism also supports *variable declarations* and *facts*. A variable declaration specifies the type(s) of a variable. A fact has the same format as a condition/conclusion of a rule, however, it can have as value the special symbol "unknown". Facts represent either initial conditions or intermediate/final conclusions produced during an inference course.

3 KNOWLEDGE BASE CONSTRUCTION

There are two ways of constructing a hybrid rule base (HRB), a direct and an indirect. The direct method is the normal way of constructing a HRB. The indirect method can be used as well, if it is more convenient to construct the initial knowledge base using symbolic rules.

3.1 Direct method

3.1.1 Constructing and training neurules

In the direct method both types of rules are used. Symbolic rules are typically used to represent conclusions produced in a unique and exact way or conclusions that cannot be represented by a single neurule (see subsequent sections). Neurules are used in all other cases.

In constructing a neurule, all conditions that contribute in drawing a conclusion constitute the inputs of a rule and the conclusion its output. Thus, one has to produce as many rules as the different conclusions, intermediate or final, to be drawn. For example, in the medical diagnosis domain, if there are four symptoms expressed as the conditions C_1, C_2, C_3, C_4 and two diseases D_1, D_2, such that C_1, C_2, C_3 are involved in diagnosing D_1, and C_3, C_4 in diagnosing D_2, the following rules are constructed: "(0) **if** C_1 (0) **,** C_2 (0) **,** C_3 (0) **then** D_1", "(0) **if** C_3 (0) **,** C_4 (0) **then** D_2". Because the LMS algorithm (see next paragraph) needs no specific initial values to calculate the factors, a zero initial value is assigned to each factor by default.

After the above has been done, each neurule is individually trained via a *training mechanism*. First, the (final) values of the factors are determined. To this end, a number of training patterns, called the *training set*, are supplied for each rule. The standard least mean square (LMS) learning algorithm (see e.g. [3]) is employed to calculate the factors. However, in cases where the mechanism fails to find factors satisfying all training patterns, symbolic rules are employed (see also Section 3.2.2). After training, reorganisation takes place.

3.1.2 Reorganising neurules

We distinguish between two types of neurules in a HRB, negative-bias and positive-bias rules. A neurule is a *negative-bias rule* if it has a negative bias factor ($sf_0 < 0$), whereas it is a *positive-bias rule* if it has a positive bias factor ($sf_0 > 0$). Also, 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* according to the values of their significance factors. Also, the conditions in its positive group are put in front of those in its negative one. The rationale behind all this orderings will become clear in Section 4.2.

Additionally, for each neurule, a *critical condition* is determined:

1. **Definition 1.** Let $\{C_1,..., C_n\}$ the positive group of a negative-bias rule ($sf_0 < 0$). Condition C_{cr} ($1 \le cr \le n$) is its critical condition iff
$$\text{CRIT} = \sum_{i=cr+1}^{n} sf_i \le - sf_0.$$
2. **Definition 2.** Let $\{C_1,..., C_n\}$ the negative group of a positive-bias rule ($sf_0 > 0$). Condition C_{cr} ($1 \le cr \le n$) is its critical condition iff
$$\text{CRIT} = \sum_{i=cr+1}^{n} sf_i \ge sf_0.$$

3.2 Indirect Method

3.2.1 Merging symbolic rules

In the indirect method, the knowledge base is initially constructed using only symbolic rules, as in conventional rule-based systems. Then, symbolic rules are transformed into neurules via a *conversion mechanism*. Symbolic rules with the same conclusion are typically merged into one neurule. For example, the rules R_5:"**if** C1**,** C2 **then** D" and R_6:"**if** C1**,** C3 **then** D" are intermediately transformed into "(0) **if** C1 (0)**,** C2 (0)**,** C3 (0) **then** D", which, after training, results in the neurule R_{56}:"(-2.5) **if** C1 (2)**,** C2 (1)**,** C3 (1) **then** D". Each neurule is then reorganized as in the direct method. The neurules can be retrained in a later time.

The training process is the same as in the direct method. The training set of a neurule, however, is not given, but is determined by selecting rows from the truth table of the *combined logical function* of the merged rules. The combined function

represents the disjunction of the conjunctions of the conditions of the rules. For example, the factors in the above example have been determined via the CM, with T= ([1 0 0 -1], [1 1 0 1], [0 1 1 -1], [1 0 1 1]) as the training set, which includes the necessary rows from the truth table of ((C_1 AND C_2) OR (C_1 AND C_3)) ≡ (C_1 AND (C_2 OR C_3)), the combined function of the two symbolic rules. (Although the combined truth table includes eight rows, the above four rows subsume the rest ones).

However, not all of the rows in the combined truth table are valid, due to domain-specific reasons. To this end, we first introduce the following notion:

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

We then introduce the following *invalid-row criteria*:

- related is-conditions (resp. isnot-conditions) (e.g. “fever is high” and “fever is low”) cannot be simultaneously true (resp. false).
- related is-conditions (resp. isnot-conditions) 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 the following row-remove criterion, which cannot literally detected, that is it requires 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.

3.2.2 The non-separability problem

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 a function [3].

To overcome this problem, we successively split the corresponding set of merging rules into subsets until the right factors are determined. Splitting is made in such a way that the rules in each subset have as more common or related conditions as possible and the subsets are of comparable size. So, two or more neurules may be produced. In such a situation, a subset may contain just one symbolic rule, which remains as it is. Thus, an initial set of merging rules with a non-separable training set will produce more than one neurule and possibly one or more symbolic rules.

4 THE HYBRID INFERENCE MECHANISM

4.1 Basic Process

The *hybrid inference mechanism* is based on a backward chaining strategy. There are two stacks used, a *goal stack (GS)*, where the *current goal* to be matched is always on its top, and a *rule stack (RS)*, where the *current rule* under evaluation is always on its top. Our conflict resolution strategy is based on textual order, at the moment. 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 working memory (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 the value "unknown" as its value. A condition cannot be evaluated if there is no fact in the WM with the same variable. Furthermore, it evaluates to 'false', if additionally there is no matching rule in the HRB.

4.2 Inference Heuristics

4.2.1 Incremental activation computation

To increase inference efficiency, a number of heuristics are used. First, the activation value is *incrementally computed*, that is contribution of each condition to the weighted sum is added immediately after its evaluation. As soon as the sum exceeds the threshold, computation stops.

4.2.2 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 factors in the negative group are evaluated and then the factors in the positive group. Thus, after evaluation of the conditions in the negative group has been completed, as soon as the result exceeds the threshold (0), evaluation stops and the output gets the value '1' ('true'). 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', speeding up the computation. A positive-bias rule is evaluated in a complementary way.

4.2.3 Critical condition situation

The following findings concerning the critical condition of a neurule, that can be easily proved, are applied (proofs are trivial).

- If the critical condition in a negative-bias neurule fails ($C_{cr} = 0$), provided that all preceding ones (negative and positive) also failed, the rule also fails (D = -1).
- If the critical condition in a positive-bias neurule fails ($C_{cr} = 0$), provided that all preceding ones (positive and negative) also failed, the rule succeeds (D = 1).

The first means that even if all consequent conditions succeed the activation value cannot exceed the threshold (0). Thus, under such a situation evaluation stops. The second means that even if all consequent conditions succeed the activation value cannot become less than the threshold (0). Thus, under such a situation evaluation stops.

5 AN EXAMPLE

In this section we present a simple example that illustrates how inference is performed in our hybrid formalism. Also it shows how the hybrid approach is more efficient than the plain symbolic one. To this end, a symbolic and its equivalent hybrid knowledge base are used.

Let consider the following symbolic rules. 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 we have the following initial WM: {C_1, C_3, C_5, C_6}. Our (initial) goal is D. The symbolic inference steps to prove D are illustrated in Fig.3 (left side, top-down), where a solid arrow means "puts on" the target stack, and a dashed arrow means "evaluates to". The equivalent HRB is the following: R_{12}: "(-2.5) **if** C_1 (2)* **,** C_2 (1) **,** C_3 (1) **then** D_1", R_{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 integrated inference steps to prove D are illustrated in Figures 3-4 (top-down). It is clear from the example, that plain symbolic representation is more expensive than the hybrid one. Also, notice that, due to embedded heuristics, hybrid inference is insensitive to changes to the order of the rules.

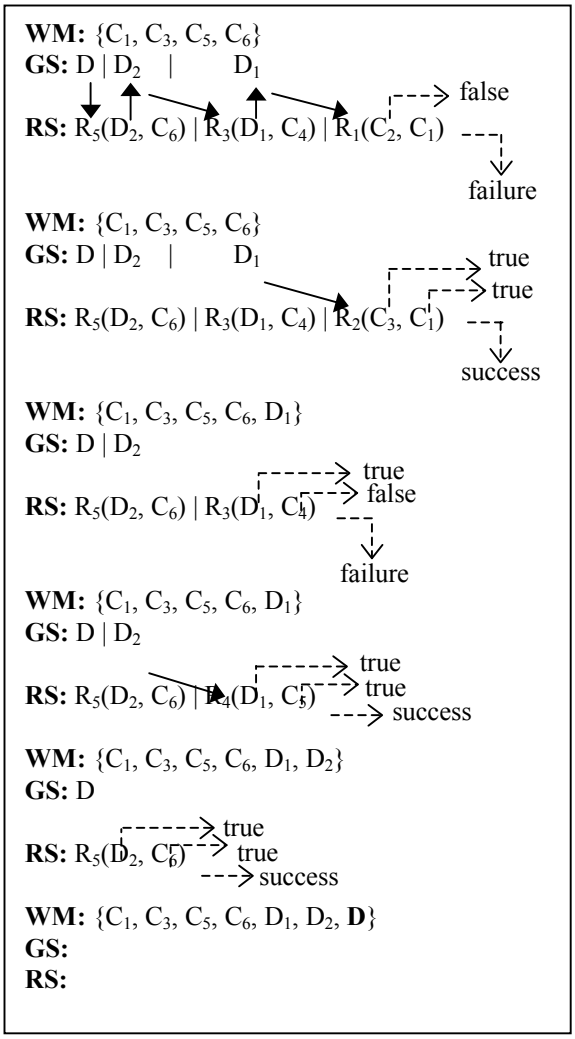

Figure 3. Symbolic Inference

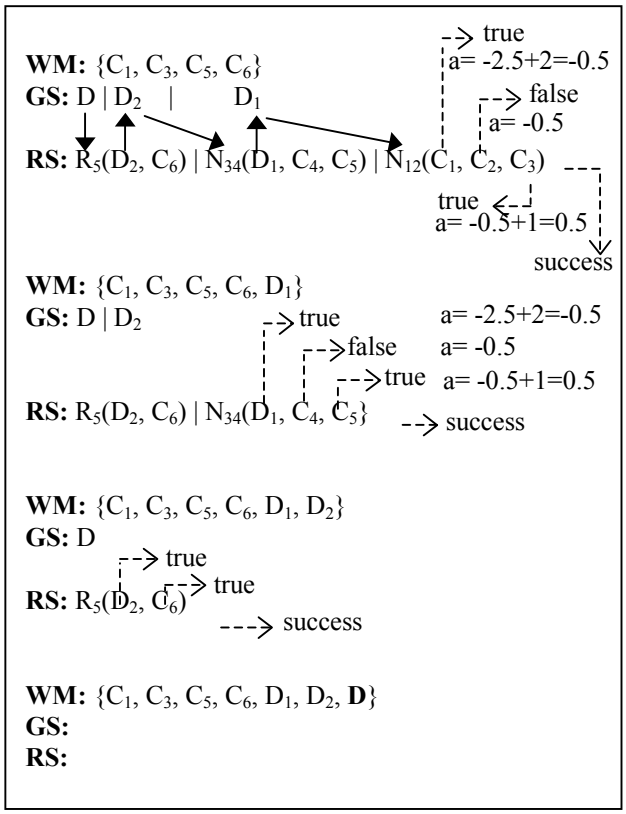

Figure 4. Integrated Inference

6 EXPERIMENTAL RESULTS

A number of experiments were made to evaluate some aspects of the hybrid formalism. The indirect method was applied to a symbolic medical rule base that contained sixty-two symbolic rules. Twenty of those rules had a unique conclusion and thus remained symbolic. From the rest forty-two rules twelve neurules and five symbolic rules were resulted. Finally, the total number of rules in the hybrid rule base was thirty-seven.

Inferences were proved to be equivalent in both cases: we had the same conclusions for the same variable-value data both in the symbolic and the hybrid case. Equivalence is basically guaranteed by the fact that neurules are trained with data

from the combined truth table of the merging rules. Also, inferences from the initial symbolic rule base were proved to be longer, in terms of the rules visited, and more expensive, in terms of the conditions evaluated, than the corresponding ones from the hybrid rule base. In the table below a number of experimental results are presented.

Table 1. Experimental Results

Inference No	Rules Visited		Conditions Evaluated		Decision
	Symbolic	Hybrid	Symbolic	Hybrid	
1	7	7	18	19	Inflammation
2	10	8	27	27	Inflammation
3	14	9	27	24	Inflammation
4	24	15	42	48	Arthritis
5	27	11	64	44	Prim. Malignant
6	27	14	70	58	Prim. Malignant
7	35	15	67	60	Prim. Malignant
8	45	23	73	61	Dec. Metabolical
9	62	25	93	89	Second. Malignant
Total	242	127	481	430	

There is an average of 47,5% reduction in the rules visited and a 10,6% reduction in the conditions evaluated in the hybrid case.

7 CONCLUSIONS

In this paper a hybrid KR formalism that integrates production rules and an ANN is presented. The adaline unit is used within the symbolic framework of production rules. So, various benefits of symbolic representation, such as naturalness and modularity, are retained. Moreover, imprecise relations between concepts can be easily represented.

Efficiency is also increased, due to the following reasons. First, the number of the rules in the knowledge base is reduced. Second, a number of inference heuristics are introduced.

The adaline unit and the LMS algorithm are not of the most powerful existing mechanisms. This is a weak point of the formalism and a good reason for further investigation.

REFERENCES

1. B. Boutsinas and M. N. Vrahatis, Nonmonotonic Connectionist Expert Systems, *Proceedings of the 2nd IMACS CSC'98* (Athens, 1998).
2. Fu L-M and L-C Fu, Mapping rule-based systems into neural architecture, *Knowledge-Based Systems* **3** (1990) pp. 48-56.
3. Gallant S.I., Neural Network Learning and Expert Systems (MIT Press, 1993).
4. Ghalwash A. Z., A Recency Inference Engine for Connectionist Knowledge Bases, *Applied Intelligence* **9** (1998) pp. 201-215.
5. Hilario M., An Overview of Strategies for Neurosymbolic Integration. In *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches*, ed. by Sun R. and E. Alexandre (Lawrence Erlbaum, 1997), chapter 2.
6. Medsker L.R., Hybrid Neural Networks and Expert Systems (Kluwer Academic Publishers, Boston, 1994).
7. Towell G. G. and Shalvik J. W., Knowledge-based artificial neural networks, *Artificial Intelligence* **70** (1994) pp. 119-165.
8. Sun R., Integrating Rules and Connectionism for Robust Commonsense Reasoning (Sixth-Generation Computer Technology, John Wiley & Sons, 1994).
9. Sun R. and E. Alexandre (Eds), Connectionist-Symbolic Integration: From Unified to Hybrid Approaches (Lawrence Erlbaum, 1997).