Published in the Proceedings of the 8th Panhellenic Conference on Informatics, Nicosia, Cyprus (November 2001) Vol.1 422-431

HYMES: A HYbrid Modular Expert System with Efficient Inference and Explanation

Ioannis Hatzilygeroudis and Jim Prentzas*

University of Patras, School of Engineering
Dept of Computer Engin. & Informatics, 26500 Patras, Hellas (Greece)
&
Computer Technology Institute, P.O. Box 1122, 26110 Patras, Hellas (Greece)
E-mails: ihatz@cti.gr, ihatz@ceid.upatras.gr, prentzas@ceid.upatras.gr
Phone: +30-61-960321, Fax: +30-61-960322
URL: http://mmlab.ceid.upatras.gr/aigroup/ihatz/

Abstract

A HYbrid Modular Expert System, called HYMES, is presented. HYMES provides a dual representation scheme: a symbolic one, based on conventional symbolic rules, and a hybrid one, based on neurules, a kind of rules that combine a symbolic and a connectionist representation. Symbolic rules are internally converted into neurules, for efficiency reasons. In this way, hybrid modular knowledge bases can be constructed. Also, both representation schemes share a common inference engine. The inference engine is based on a connectionist technique, which is proved to be quite efficient. Furthermore, HYMES possesses an efficient and natural explanation mechanism.

1. Introduction

Constructing conventional expert systems greatly depends on the availability of experts and their ability to make explicit their way of reasoning, which is a hard task. So, knowledge acquisition from experts is usually referred to as a bottleneck (Medsker 1994). On the other hand, constructing a neural network mostly depends on the availability of empirical data (examples). This makes construction of neural networks easy. Additionally, neural networks are very efficient in computing their outputs. However, they lack explanation capabilities, in contrast to expert systems, especially the rule-based ones. This makes them look like black boxes, unnatural and not understandable.

Therefore, there have been efforts at combining expert systems and neural networks into hybrid systems, in order to exploit their benefits. In some of them, called embedded systems, a neural network is used as the inference engine of an expert system. For example, in NEULA (Tirri 1991, Tirri 1995) a neural network selects the next rule to fire. Similarly, SETHEO (Letz et al 1992) uses a neural network to select the most promising node at each inference step in its theorem proving process. Also, LAM (Medsker 1994) uses two neural networks as partial problem solvers. However, those systems exploit only the efficiency of neural networks, leaving out aspects such as explanation, naturalness and modularity.

* Research partially supported by GSRT of Greece, Project No ΕΔ234, Program ΠΕΝΕΔ'99.

On the other hand, connectionist expert systems are integrated systems that represent relationships between concepts, considered as nodes in a neural network. Weights are set in a way that makes the network infer correctly. MACIE (Gallant 1988, Gallant 1993) is such a system. Two characteristics of MACIE are: its ability to reason from partial data and its ability to provide explanations in the form of if-then rules. To improve performance of connectionist expert systems, the "recency inference engine" is introduced in (Ghalwash 1998). In order to assess its performance, the 'convergence rate' is used, which is based on the number of known and necessary (required) inputs. However, this does not take into account the number of internal computations, which for large networks may be of importance.

A weak point of connectionist expert systems is that they absolutely rely on empirical data. So, they are useless in cases where such data is not available. That is, their "hybridness" invalidates the symbolic component representation scheme.

In this paper, we present HYMES, a Hybrid Modular Expert System tool for constructing expert systems. HYMES provides two separate, although closely related, representation schemes, a symbolic and a neurosymbolic. The symbolic representation scheme is based on symbolic, if-then rules, whereas the neurosymbolic one on *neurules* (Hatzilygeroudis and Prentzas 2000a, Hatzilygeroudis and Prentzas 2000b), a hybrid rule-based representation scheme. However, symbolic rules are converted into neurules, so that a common inference engine is used. An inference strategy and an explanation mechanism are provided.

The structure of the paper is as follows. Sections 2 and 3 present the knowledge representation schemes and the architecture of HYMES respectively. In Sections 4 and 5 the hybrid inference strategy and the explanation mechanism are presented respectively. Section 6 presents some examples, whereas Section 7 experimental results. Finally, Section 8 concludes.

2. Knowledge Representation in HYMES

Neurules (: *neur*al *rules*) are a kind of hybrid rules. Each neurule (Fig. 1a) is considered as an adaline unit (Fig.1b). 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 *bias* of the unit.

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 formulas:

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

where **a** is the *activation value* and *f(x)* the *activation function*, which is a threshold function:

$$f(\mathbf{a}) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Hence, the output can take one of two values, '-1' and '1', representing failure and success of the rule respectively.

The general syntax (structure) of a neurule is (where '{ }' denotes zero, one or more occurrences and '< >' denotes non-terminal symbols):

<rule>::= (<bias-factor>) **if** <conditions>
then <conclusions>
<conditions>::= <condition> {**,** <condition>}
<conclusions>::= <conclusion> {**,** <conclusion>}
<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. A variable in a condition can be either an *input variable* or an *intermediate variable*, whereas a variable in a conclusion can be either an *intermediate* or an *output variable* or both. An input variable takes values from the user (input data), whereas intermediate and output variables take values through inference, since they represent intermediate and final conclusions respectively.

<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* or a *number*. The significance factor of a condition represents the significance (weight) of the condition in drawing the conclusion(s).

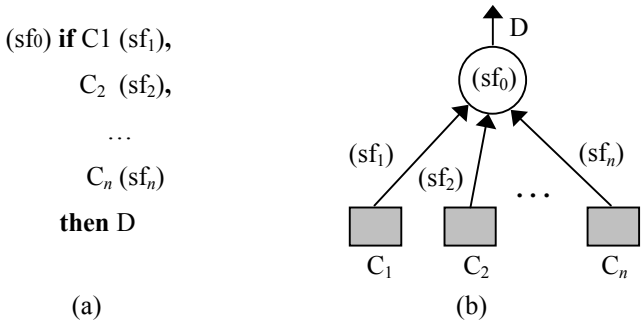

Fig.1 (a) Form of a neurule (b) corresponding adaline unit

Neurules are distinguished in *intermediate* and *output rules*, depending on whether their conclusions contain intermediate or output variables respectively.

Symbolic rules have the same syntax and semantics as neurules, except of the bias and significance factors, which are missing. Symbolic rules are finally converted into neurules. (For example neurules see Section 6.1).

3. The Architecture of HYMES

In Fig.2, the architecture of HYMES is illustrated. The run-time system (in the dashed rectangle) consists of four modules, functionally similar to those of a conventional rule-based system: the *hybrid rule base (HRB)*, the *hybrid inference engine (HIE)*, the *working memory (WM)* and the *explanation mechanism (EXM)*.

The HRB contains neurules produced either from empirical (training) data or from symbolic rules. In the first case, the initial neurules, constructed by the user using dependency information between concepts (variables), are trained by the *neurules training mechanism (NTM)* using training examples produced from available empirical data by the user. This process is described in (Hatzilygeroudis and Prentzas 2001). The user is assisted in the creation of training data by the *training data elicitation mechanism (TDEM)* described in section 3.1. In the second case, the symbolic rules, constructed by the user via a knowledge acquisition process, are converted to neurules via the *conversion and training mechanism (CTM)*. The conversion process is described in (Hatzilygeroudis and Prentzas 2000b). That process may produce not only neurules, but also symbolic rules due to the fact that some symbolic rules may correspond to non-separable (boolean) functions. However, single symbolic rules can be easily converted into neurules: we set all $sf_i = 1$ and $sf_0 = -(n - 0.5)$, where n is the number of conditions (inputs) of the rule (neural unit). For inference reasons (see Section 4), the conditions in each neurule are ranked in descending order, according to the absolute values of their significance factors.

The HIE is responsible for making inferences by taking into account the input data in the WM and the rules in the HRB. The WM contains *facts*. A fact has the same format as a condition or a conclusion of a rule. However, it can have as value the special symbol "unknown". Facts represent either initial conditions or intermediate or final conclusions produced during an inference course. Finally, the EXM is responsible for producing an explanation of the reasoning the system used to reach a conclusion, in the form of if-then rules.

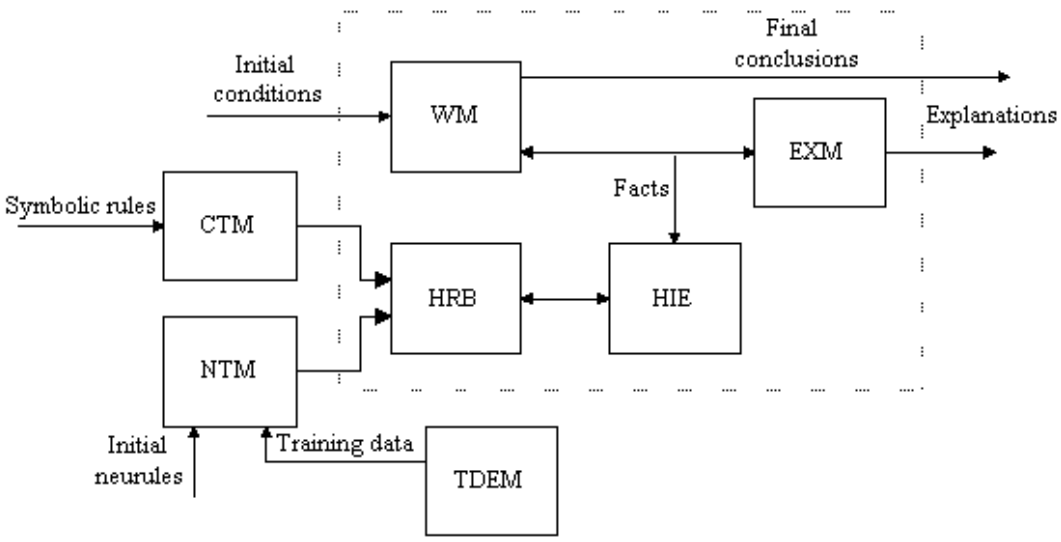

Fig. 2. The architecture of HYMES

4. The Hybrid Inference Engine

The *hybrid inference engine* implements the way neurules co-operate to reach a conclusion. It is based on the 'firing ratio', a measurement of the firing intention of a neurule, which is similar to the 'convergence ratio', introduced in (Ghalwash 1998).

Generally, the output of a neurule is computed according to Eq. (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)* as follows:

$$kn-sum = sf_0 + \sum_{cond_i \in E} sf_i * C_i \quad (2)$$

$$rem-sum = \sum_{cond_i \in U} |sf_i| \quad (3)$$

where E is the set of evaluated conditions, U the set of unevaluated conditions and C_i is the value of condition $cond_i$. So, 'known-sum' is the weighted sum of the values of the already known (i.e. evaluated) conditions (inputs) of the corresponding neurule and '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 $kn\text{-}sum > 0$. So, we define the *firing ratio (fr)* of a neurule as follows:

$$fr = \frac{|kn\text{-}sum|}{rem\text{-}sum} \quad (4)$$

The firing ratio of a neurule is an estimate of its intention that its output will become '±1'. Whenever $fr > 1$, the values of the evaluated conditions can determine the value of its output, regardless of the values of the unevaluated conditions. The rule then evaluates to '1' (true), if $kn\text{-}sum > 0$ or to '-1' (false), if $kn\text{-}sum < 0$. In the first case, we say that the neurule is *fired*, whereas in the second that it is *blocked*. Notice, that the firing ratio has meaning only if $rem\text{-}sum \neq 0$. If $rem\text{-}sum = 0$, all the conditions have been evaluated and its output is evaluated according to *kn-sum*.

Initially, the values of the variables (conditions) may be not known to the system. The *kn-sum* for every neurule is then equal to its bias factor, whereas its *rem-sum* is equal to the sum of the absolute values of all its significance factors. If the value of a variable becomes known, it influences the values of the conditions containing it and hence the known sums, the remaining sums and the firing ratios of the corresponding unevaluated neurules. If an intermediate neurule evaluates, the known sums, the remaining sums and the firing ratios of all other unevaluated neurules with a condition containing its intermediate variable should be also updated. Obviously, a firing ratio is updated only if the corresponding remaining sum is not equal to zero.

Unevaluated neurules that are updated, after a variable value propagation, constitute the *participating* neurules. The inference mechanism tries then to focus on participating neurules whose firing ratio is close to exceeding '1'. More specifically, it selects the one with the maximum firing ratio, because it is the most likely, it intends, to fire. The system tries to evaluate the first unevaluated condition, which is the one with the maximum absolute significance factor (recall that conditions are sorted). After evaluation of the condition, *kn-*

sum, rem-sum and *fr* are computed. If *rem-sum* = 0 or *fr* > 1, it evaluates and its conclusion is put in the WM. If the system reaches a final conclusion, it stops.

More formally, the inference algorithm is as follows:

1. If there is input data in the WM, update the known sums, the remaining sums and the firing ratios of all the affected neurules, which become the participating neurules. If there is no initial input data, compute the initial values of the known sums, the remaining sums and the firing ratios of all the neurules and regard them as participating.
2. If there is a participating neurule with (*fr* > 1 or *rem-sum* = 0) and (*kn-sum* > 0), mark the rule as fired and its conclusion as 'true' and put it in the WM. If there is a participating neurule with (*fr* > 1 or *rem-sum* = 0) and (*kn-sum* < 0), mark the rule as blocked and if there are no other non-blocked neurules with the same conclusion, mark its conclusion as 'false' and put it in the WM. Remove it from the participating rules. Update all the affected conditions as well as the known sums, the remaining sums and the firing ratios of the corresponding neurules and put them in the participating neurules. Execute this step recursively.
3. While the system has not reached a final conclusion do:
   - 3.2.1 From the participating neurules select the one with the maximum firing ratio. If there are no participating neurules, select the unevaluated with the maximum *fr*.
   - 3.2.2 Consider the first unevaluated condition of the selected neurule. If it contains an input variable, ask the user for its value. If it contains an intermediate variable instead, find an unevaluated neurule with the maximum firing ratio that contains the variable in its conclusion and execute this step recursively taking this neurule as the selected.
   - 3.2.3 Clear participating rules. According to the input data given by the user, update all the affected conditions as well as the known sums, the remaining sums and the firing ratios of the corresponding neurules and put them in the participating neurules.
   - 3.2.4 The same as 2.
4. If there are no conclusions in the WM containing output variables, stop (failure). Otherwise, display the conclusions and stop (success).

5. The Explanation Mechanism

The explanation mechanism justifies inferences by producing a set of if-then rules, explaining how the conclusions were reached. The conclusions of the explanation rules contain the inferred output variables. Their conditions contain a subset of the input and intermediate variables participating in drawing the conclusions, that is those variables whose values were either given by the user or inferred during the inference process, possibly with changes to their predicates. More specifically, the conditions in the explanation rules are the ones with the most positive contribution in producing the output of the corresponding neurule.

For explanation purposes, the conditions of an evaluated neurule are distinguished in positive and negative conditions. In the case of a fired neurule, *positive conditions* are either

the ones evaluated to true ('1') and had a positive significance factor or the ones evaluated to false ('-1') and had a negative significance factor. In the case of a blocked neurule, positive conditions are either the ones evaluated to true ('1') and had a negative significance factor or the ones evaluated to false ('-1') and had a positive significance factor. Conditions that are unknown or negative are not included in explanation rules. Furthermore, some of the positive conditions may be also not included, based on the fact that they are not necessary.

The following process is followed in order to find the necessary positive conditions:

1. Let *pos-sum* be the weighted sum of all the positive conditions plus the bias factor, *neg-sum* be the weigthed sum of the negative conditions and *nec-set*, *unnec-set* the sets of necessary and unnecessary positive conditions respectively. Set *nec-set* to include all positive conditions and *unnec-set* to be the empty set.
2. While $|pos\text{-}sum|>|neg\text{-}sum|$ and *nec-set* contains more than one condition do:
   2.1. Let C_i be the positive condition with the least absolute significance factor, say sf_i. Delete C_i from *nec-set* and insert it to *unnec-set*.
   2.2. $pos\text{-}sum \leftarrow pos\text{-}sum - C_i * sf_i$
   $neg\text{-}sum \leftarrow neg\text{-}sum + |sf_i|$
3. The necessary positive conditions are those in *nec-set* plus the last one inserted to *unnec-set*.
4. If a condition in the *nec-set* is evaluated to false, change its predicate from ‘is’ to ‘isnot’ and vice versa.

Following is a more formal outline of the explanation mechanism:

For each of the fired output neurules do:

1. Generate an if-then rule whose conclusion is the neurule's conclusion and its conditions are the necessary positive conditions of the neurule. Make possible changes to the predicates according to the values of the conditions (as in step 4, above).
2. For each condition containing an intermediate variable, an if-then rule is produced based on an evaluated neurule having that condition as its conclusion. The predicate of the condition is the one that was before its possible change from 'is' to 'is-not' and vice versa took place. More than one neurule may have the condition as their conclusion. If the condition was evaluated to true, only one of these neurules will have its output evaluated to '1' and will be the chosen one. If the condition was evaluated to false, then the outputs of all neurules having it as their conclusion were evaluated to '-1'. In this case, the neurule with the maximum firing ratio will be chosen. Step 1 is recursively executed for the chosen neurule.

6. Examples

6.1 An example knowledge base

We use as an example to illustrate the functionalities of our system the one presented in (Gallant 1993). It contains training data dealing with acute theoretical diseases of the sarcophagus. There are six symptoms (Swollen feet, Red ears, Hair loss, Dizziness, Sensitive aretha, Placibin allergy), two diseases (Supercilliosis, Namastosis) whose diagnoses are

based on the symptoms and three possible treatments (Placibin, Biramibio, Posiboost). Also, dependency information is provided. We used the dependency information to construct the initial neurules and the training data provided to train them. The produced knowledge base, which contains six neurules (DR1-DR6), is illustrated in Fig.3. An equivalent knowledge base forming a multilevel network is presented in (Gallant 1988; Gallant 1993).

6.2 An example inference

We suppose that there is the following initial data in the WM: 'HairLoss is true'. So, the firing ratios of neurules DR1, DR2 and DR4 should be updated. (We notice here that the initial firing ratios of the neurules are calculated at construction time). The inference tracing is presented below.

WM: {'HairLoss is true' (TRUE)}
Affected neurules: DR1, DR2, DR4
Updated *frs*: |3.2/4.4| = 0.73 (DR1), |3.2/6.4| = 0.5 (DR2), |-7.6/6.4| = 1.19 (DR4)
Fired neurules:
Blocked neurules: DR4
Participating neurules: [DR1, DR2]

DR1: (-0.4) **if** RedEars is true (-0.8),
SwollenFeet is true (3.6),
HairLoss is true (3.6)
then Disease is Supercilliosis

DR2: (1.4) **if** Dizziness is true (4.6),
SensitiveAretha is true (1.8),
HairLoss is true (1.8)
then Disease is Namastosis
DR3: (-2.2) **if** PlacibinAllergy is true (-5.4),
Disease is Supercilliosis (4.6)
Disease is Namastosis (1.8),
then Treatment is Placibin

DR4: (-4.0) **if** HairLoss is true (-3.6),
Disease is Namastosis (3.6),
Disease is Supercilliosis (2.8)
then Treatment is Biramibio

DR5: (-2.2) **if** Treatment is Biramibio (-2.6),
Treatment is Placibin (1.8)
then Treatment is Posiboost

DR6: (-2.2) **if** Treatment is Placibin (-1.8),
Treatment is Biramibio (1.0)
then Treatment is Posiboost

Fig. 3. An example knowledge base.

Given that there is a neurule (DR4) with *fr* > 1, we should propagate its conclusion (intermediate data).

Intermediate data: 'Treatment is Biramibio' (FALSE)
WM: {'HairLoss is true' (TRUE), 'Treatment is Biramibio' (FALSE)}
Affected neurules: DR5, DR6
Updated *fr*s: |0.4/1.8| = 0.22 (DR5), |-3.2/1.8| = 1.78 (DR6)
Fired neurules:
Blocked neurules: DR4, DR6
Participating neurules: [DR5, DR1, DR2]

Although there is a neurule (DR6) with *fr* > 1, we do not propagate its conclusion, because it is an output neurule. Also, given that there is another non-blocked rule with the same conclusion, we do not put the conclusion in the WM. So, we proceed by selecting a neurule from the participating set.

Selected neurule: DR1 (the maximum *fr*)
User data: 'SwollenFeet is true' (FALSE)
WM: {'HairLoss is true' (TRUE), 'Treatment is Biramibio' (FALSE), 'SwollenFeet is true' (FALSE)}
Affected neurules: DR1
Updated *fr*s: |-0.4/0.8| = 0.5 (DR1)
Participating neurules: [DR1]
Fired neurules:
Blocked neurules: DR6, DR4

Given that there is no neurule with *fr* > 1, we proceed with a participating rule.

Selected neurule: DR1 (the maximum *fr*)
User data: 'RedEars is true' (FALSE)
WM: {'HairLoss is true' (TRUE), 'Treatment is Biramibio' (FALSE), 'SwollenFeet is true' (FALSE), 'RedEars is true' (FALSE)}
Affected neurules: DR1
Updated *fr*s: *kn-sum* = 0.4, *rem-sum* = 0 (because all conditions of DR1 have been evaluated, *fr* has no meaning)
Participating neurules:
Fired neurules: DR1
Blocked neurules: DR6, DR4

Given that *kn-sum* > 0 and DR1 is fired, its conclusion should be propagated.

Intermediate data: 'Disease is Supercilliosis' (TRUE)
WM: {'HairLoss is true' (TRUE), 'Treatment is Biramibio' (FALSE), 'SwollenFeet is true' (FALSE), 'RedEars is true' (FALSE), 'Disease is Supercilliosis' (TRUE)}
Affected neurules: DR3 (DR4 has been evaluated)
Updated *fr*s: |2.4/7.2| = 0.33 (DR3)
Participating neurules: DR3

Fired neurules: DR1
Blocked neurules: DR6, DR4

Given that there is no neurule with *fr* > 1, we proceed.

Selected neurule: DR3
User data: 'PlacibinAllergy is true' (FALSE)
WM: {'HairLoss is true' (TRUE), 'Treatment is Biramibio' (FALSE), 'SwollenFeet is true' (FALSE), 'RedEars is true' (FALSE), 'Disease is Supercilliosis' (TRUE), 'PlacibinAllergy is true' (FALSE)}
Affected neurules: DR3
Updated *frs*: |7.8/1.8| = 4.33 (DR3)
Participating neurules:
Fired neurules: DR1, DR3
Blocked neurules: DR6, DR4

Given that there is a neurule (DR3) with *fr* > 1 and its conclusion is both an output and intermediate we should do both, put it in the WM and propagate it.

Output data: 'Treatment is Placibin' (TRUE)
Intermediate data: 'Treatment is Placibin' (TRUE)
WM: {'HairLoss is true' (TRUE), 'Treatment is Biramibio' (FALSE), 'SwollenFeet is true' (FALSE), 'RedEars is true' (FALSE), 'Disease is Supercilliosis' (TRUE), 'PlacibinAllergy is true' (FALSE), 'Treatment is Placibin' (TRUE)}
Affected neurules: DR5
Updated *frs*: *kn-sum* = 2.2, *rem-sum* = 0 (because all conditions of DR5 have been evaluated, *fr* has no meaning)
Participating neurules:
Fired neurules: DR5, DR3, DR1
Blocked neurules: DR6, DR4

Given that *kn-sum* > 0 and the DR5 is fired and it is an output rule, its conclusion is put in the WM.

Output data: 'Treatment is Posiboost' (TRUE)
Participating neurules:
WM: {'HairLoss is true' (TRUE), 'Treatment is Biramibio' (FALSE), 'SwollenFeet is true' (FALSE), 'RedEars is true' (FALSE), 'Disease is Supercilliosis' (TRUE), 'PlacibinAllergy is true' (FALSE), 'Treatment is Placibin' (TRUE), 'Treatment is Posiboost' (TRUE)}
Participating neurules:
Fired neurules: DR5, DR3, DR1
Blocked neurules: DR6, DR4

So, finally we have the following final conclusions.

Output data: 'Treatment is Placibin', 'Treatment is Posiboost'

6.3 An explanation example

In this section, we present the explanation that will be produced by our system for the above inference. We give tracings for the first two of them. Given that there are two outputs, 'Treatment is Posiboost' and 'Treatment is Placibin', the explanation mechanism should provide explanations for them. In order to generate an explanation rule for the first output, the explanation mechanism will examine neurule DR5 whose output is 'Treatment is Posiboost' and was evaluated to '1' (TRUE). The explanation rule extracted is the following:

EXR1
if Treatment isnot Biramibio,
Treatment is Placibin
then Treatment is Posiboost.

The tracing of the generation of the above explanation rule is as follows:

nec-set: {'Treatment is Biramibio', 'Treatment is Placibin'}
unnec-set: { }
pos-sum: -2.2 + 2.6 + 1.8 = 2.2
neg-sum: 0

Since |pos-sum| > |neg-sum|:

nec-set: {'Treatment is Biramibio'}
unnec-set: {'Treatment is Placibin'}
pos-sum: 2.2 - 1.8 = 0.4
neg-sum: 1.8

Since |pos-sum| < |neg-sum|:

nec-set: {'Treatment is Biramibio', 'Treatment is Placibin'}.

Due to the fact that 'Treatment is Biramibio' was evaluated to false, we change its predicate from 'is' to 'isnot'.

Because the explanation rule EXR1 contains an intermediate variable, a corresponding explanation rule should be generated for it (with conclusion 'Treatment isnot Biramibio'). To this end, neurule DR4 is examined and the following explanation rule is generated:

EXR2
if HairLoss is true
then Treatment isnot Biramibio

The tracing of the generation of the above explanation rule is as follows:

nec-set: {'HairLoss is true'}
unnec-set: { }

pos-sum: -4.0 -3.6 = -7.6
neg-sum: 2.8

Since nec-set has only one condition, it remains as it is. Furthermore, since DR4 was evaluated to be FALSE, we change the predicate of its conclusion from 'is' to 'isnot'.

For the output 'Treatment is Placibin', neurule DR3 is examined and the following explanation rule is generated:

EXR3
if PlacibinAllergy is true,
  Disease is Supercilliosis
then Treatment is Placibin

Given that there is again an intermediate variable, one more explanation rule for its condition ('Disease is Supercilliosis') should be provided. This time neurule DR1 is examined and the following explanation rule is generated:

EXR4
if HairLoss is true,
  RedEars is false
then Disease is Supercilliosis

7. Experimental Results

This section presents experimental results comparing the performance of our inference and explanation mechanism with those presented in (Ghalwash 1998). Our inference mechanism was applied to two neurule bases, based on two datasets (described below). The inference mechanism in (Ghalwash 1998) was applied to two connectionist knowledge bases, which correspond to the same datasets and were created by the technique described in (Gallant 1988, Gallant 1993). The comparison is made in terms of the number of inputs asked by the system in order to draw conclusions (as suggested in (Ghalwash 1998)) and the number of the conditions/inputs visited for some kind of computation in drawing conclusions.

The first dataset is that used for our example knowledge base (Section 6.1) and taken from (Gallant 1988; Gallant 1993). The dataset is incomplete. It consists of 8 input data patterns out of 64 possible. The second dataset is taken from the machine learning ftp repository (see Dataset2 in the References) and involves a database for fitting contact lenses. This dataset is complete and contains 24 input patterns each consisting of four input and one output attribute (variable). The input attributes are: age of the patient (young, pre-presbyopic, presbyopic), spectacle prescription (myope, hypermyope), astigmatic (no, yes), tear production rate (reduced, normal). The target attribute takes three possible values (1, 2 or 3) denoting if the patient should be fitted with hard contact lenses, soft contact lenses or if he/she should not be fitted with contact lenses. The equivalent knowledge base is forming a multilevel network constructed according to the method described in (Gallant 1988; Gallant 1993), as the one of the first data set.

Table 1 depicts the results. Initially, the values of all variables were not known. We call ACUTE and LENSES the two types of the knowledge bases. The comparison shows that our inference engine performs slightly better than the other system as far as the number of asked

variables/inputs is concerned and much better as far as conditions/inputs visits are concerned. Although this latter is not significant for small knowledge bases, it may become important for very large knowledge bases.

Table 1

KB	*INFER-ENCES*	*HYMES*			*GALLANT-GHALWASH*		
		ASKED	*EVALS*	*EXPLS*	*ASKED*	*EVALS*	*EXPLS*
ACUTE	48	166 (3.5)	486 (10.1)	59 (1.2)	167 (3.5)	738 (15.4)	71 (1.5)
LENSES	24	79 (3.3)	602 (25)	24 (1)	80 (3.3)	886 (36.9)	36 (1.5)

In table 2, the explanation rules produced by HYMES and the system in (Ghalwash 1998), for the inference in 6.2, are presented, where 'uA' and 'uB' are intermediate variables used in (Ghalwash 1998) to construct the equivalent (neural) knowledge base. The rules produced by HYMES are less and more natural. The existence of meaningless intermediate variables, in the other system, increases the number of its explanation rules and make them difficult to comprehend.

Table 2

HYMES	GHALWASH	
EXR1 **if** Treatment isnot Biramibio, Treatment is Placibin **then** Treatment is Posiboost. EXR2 **if** HairLoss is true **then** Treatment isnot Biramibio EXR3 **if** PlacibinAllergy is true, Disease is Supercilliosis **then** Treatment is Placibin EXR4 **if** HairLoss is true, RedEars is false **then** Disease is Supercilliosis	GH-EXR1 **if** uA isnot true, uB isnot true **then** Treatment is Posiboost GH-EXR2 **if** Treatment is Placibin, Treatment isnot Biramibio **then** uA isnot true GH-EXR3 **if** Treatment is Placibin, Treatment isnot Biramibio **then** uB isnot true	GH-EXR4 **if** HairLoss is true **then** Treatment isnot Biramibio GH-EXR5 **if** PlacibinAllergy is true, Disease is Supercilliosis **then** Treatment is Placibin GH-EXR6 **if** HairLoss is true, RedEars is false **then** Disease is Supercilliosis

8. Conclusion

In this paper, we presented a modular hybrid expert system tool, called HYMES, whose knowledge base consists of neurules, a type of hybrid rules integrating symbolic rules with neurocomputing. An attractive feature of the neurules is that compared to other connectionist approaches they retain the modularity and to some degree the naturalness of symbolic rules. The neurules contained in the system's knowledge base are constructed either by converting existing symbolic rules or directly from empirical data in the form of training examples.

The run-time part of the system contains an inference mechanism for drawing conclusions based on facts and an explanation mechanism for justifying the reached conclusions.

Experimental results based on two datasets have shown an improvement to the performance of the inference engine compared to another system.

Also, the explanation rules, produced by the explanation mechanism are more natural and less than the ones produced by the other system. This fact, besides the computational cost, raises an issue of comprehensibility as far as the user is concerned. The more explanation rules are presented to the user, the more confused he/she will be.

References

Dataset2. ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

Gallant, S.I. 1988. Connectionist expert systems, *Communications of the ACM,* 31(2), 152-169.

Gallant, S.I. 1993. *Neural Network Learning and Expert Systems*, MIT Press.

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

Hatzilygeroudis, I. and Prentzas, J. 2000a. Neurules: Integrating Symbolic Rules and Neurocomputing, in D. Fotiades and S. Nikolopoulos (Eds), Advances in Informatics, World Scientific Pub., 122-133.

Hatzilygeroudis, I. and Prentzas, J. 2000b. Neurules: Improving the Performance of Symbolic Rules. *International Journal on AI Tools (IJAIT)*, 9(1), 113-130.

Hatzilygeroudis, I. and Prentzas, J. 2001. Constructing Modular Hybrid Knowledge Bases for Expert Systems. *International Journal on AI Tools (IJAIT)*, 10(1-2), 87-105.

Medsker L. R. 1994, *Hybrid Neural Networks and Expert Systems*, Kluwer Academic Publishers, Boston.

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

Tirri H. 1991. Implementing Expert Systems Rule Conditions by Neural Networks. *New Generation Computing*, 10, pp. 55-71.

Tirri H. 1995. Replacing the Pattern Matcher of an Expert System with a Neural Network. In *Intelligent Hybrid Systems*, Goonatilake S. and Sukdev K. (Eds), John Wiley & Sons.