ELSEVIER

Expert Systems with Applications 27 (2004) 63–75

Expert Systems with Applications

www.elsevier.com/locate/eswa

Integrating (rules, neural networks) and cases for knowledge representation and reasoning in expert systems

Ioannis Hatzilygeroudis[a,b,*], Jim Prentzas[a,b]

[a]Department of Computer Engineering and Informatics, School of Engineering, University of Patras, 26500 Patras, Greece
[b]Research Academic Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece

Abstract

In this paper, we present an approach that integrates symbolic rules, neural networks and cases. To achieve it, we integrate a kind of hybrid rules, called neurules, with cases. Neurules integrate symbolic rules with the Adaline neural unit. In the integration, neurules are used to index cases representing their exceptions. In this way, the accuracy of the neurules is improved. On the other hand, due to neurule-based efficient inference mechanism, conclusions can be reached more efficiently. In addition, neurule-based inferences can be performed even if some of the inputs are unknown, in contrast to symbolic rule-based inferences. Furthermore, an existing symbolic rule-base with indexed exception cases can be converted into a neurule-base with corresponding indexed exception cases. Finally, empirical data can be used as a knowledge source, which facilitates knowledge acquisition. We also present a new high-level categorization of the approaches integrating rule-based and case-based reasoning.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Hybrid knowledge representation; Hybrid reasoning; Rule-based reasoning; Case indexing; Neurocomputing; Hybrid expert systems

1. Introduction

Symbolic rules constitute a popular knowledge representation scheme used in the development of expert systems. Rules represent general knowledge of the domain and exhibit a number of attractive features such as, naturalness, modularity and ease of explanation. One of their major drawbacks is the difficulty to acquire them. The traditional process of eliciting rules through interaction with the expert may turn out to be a bottleneck, causing delays in the system's overall development (Gonzalez & Dankel, 1993). Furthermore, the acquired rules may be imperfect and not covering the full complexities of the domain. Rule induction methods deal with many of these disadvantages, but may still be unable to recognize exceptions in small, low frequency sections of the domain (Cercone, An, & Chan, 1999).

Case-based reasoning offers some advantages compared to symbolic rules and other knowledge representation and reasoning formalisms. Cases represent specific knowledge of the domain, are natural and usually easy to obtain (Aamodt & Plaza, 1994; Kolodner, 1993; Leake, 1996). Incremental learning comes natural to case-based reasoning. New cases can be inserted into a knowledge base without making changes to existing knowledge. The more cases are available, the better the domain knowledge is represented. Therefore, the accuracy of a case-based system can be improved throughout its operation stage, as new cases become available. A negative aspect of cases compared to symbolic rules is that they do not provide concise representations of the incorporated knowledge. Also, it is not possible to represent heuristic knowledge. Furthermore, the time-performance of the retrieval operations is not always the desirable.

Approaches integrating rule- and case-based reasoning have resulted in interesting and effective knowledge representation schemes (Aha & Daniels, 1998; Branting, 1999; Cercone et al., 1999; Freuder, 1998; Koton, 1988; Leake, 1995; Marling, Petot, & Sterling, 1999). The objective of these efforts is to derive hybrid representations that augment the positive aspects of the integrated formalisms and simultaneously minimize their negative aspects.

* Corresponding author. Address: Department of Computer Engineering and Informatics, School of Engineering, University of Patras, 26500 Patras, Greece. Tel.: +30-2610996937; fax: +30-2610960374.

E-mail addresses: ihatz@ceid.upatras.gr, ihatz@cti.gr (I. Hatzilygeroudis); prentzas@ceid.upatras.gr (J. Prentzas).

0957-4174/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2003.12.004

However, a more interesting approach would be one integrating more than two reasoning methods towards the same objective. In this paper, we introduce an approach integrating three reasoning/computational schemes, namely rule-based reasoning, neurocomputing and case-based reasoning, in an effective way. To this end, we combine neurules (Hatzilygeroudis & Prentzas, 2000), a kind of hybrid rules, and cases, in a way similar to that in Golding and Rosenbloom (1996), where symbolic rules are combined with cases.

Neurules integrate (propositional type) symbolic rules with neurocomputing. Neurules exploit advantages from both symbolic rules and neural networks. Thus, the approach in Golding and Rosenbloom (1996) is improved in a number of ways. First, some of the benefits of the neural networks, such as knowledge acquisition from empirical data, reasoning from partial inputs and generalization capabilities, are added to the representation scheme. Second, the size of the knowledge base is significantly reduced and the performance of the approach is improved. On the other hand, neurules are also improved. Neurules can be produced from symbolic rules, which constitute their source knowledge. If source knowledge is incomplete or does not cover the full complexities of the domain, this is reflected to the produced neurules. Integrating neurules with cases, their accuracy is improved. This paper is an extension to Prentzas and Hatzilygeroudis (2002).

The rest of the paper is organized as follows. Section 2 presents neurules, whereas Section 3 presents the architecture for integrating neurule- and case-based reasoning. Section 4 presents methods for constructing the indexing scheme of the case library. Section 5 describes the hybrid inference mechanism. Section 6 presents experimental results regarding the performance of the inference process. Section 7 presents related work concerning approaches integrating rule-based with case-based reasoning. Finally, Section 8 concludes.

2. Neurules

During the last years, artificial neural networks have been used quite often in the development of expert systems (Gallant, 1993; Ghalwash, 1998). Neural networks represent a totally different approach to the problem of knowledge representation, known as connectionism (Gallant, 1988). Some advantages of neural networks are their ability to obtain their knowledge from training examples (reducing the interaction with the experts), their high level of efficiency and their ability to represent complex and imprecise knowledge.

For those reasons, recently, there has been extensive research activity at combining (or integrating) the symbolic and the connectionist approaches. More specifically, there are a number of efforts at combining symbolic rules and neural networks for knowledge representation (Fu & Fu, 1990; Towell & Shavlik, 1994). Those approaches use a neural network as a knowledge base, thus reducing knowledge elicitation from the experts to a minimum. In this way, the knowledge acquisition bottleneck can be managed. Those approaches, however, have a major drawback due to the fact that they give pre-eminence to connectionism. More specifically, the resulted knowledge bases lack the naturalness and modularity of symbolic rules. Rule extraction methods are often used in order to comprehend their encompassed knowledge (Towel & Shavlik, 1993).

Neurules are a type of hybrid rules integrating symbolic rules with neurocomputing giving pre-eminence to the symbolic component. Neurocomputing is used within the symbolic framework to improve the performance of symbolic rules (Hatzilygeroudis & Prentzas, 2000). In contrast to other hybrid approaches (Gallant, 1993; Ghalwash, 1998), the constructed knowledge base retains the modularity of production rules, since it consists of autonomous units (neurules), and also retains their naturalness in a great degree, since neurules look much like symbolic rules. The size of the produced neurule bases is significantly reduced compared to that of the equivalent symbolic rule bases. Also, the inference mechanism is a tightly integrated process, which results in more efficient inferences not only than those in symbolic rules, but also than those in connectionist approaches. Finally, explanations in the form of if–then rules can also be produced (Hatzilygeroudis & Prentzas, 2001a).

2.1. Syntax and semantics

The form of a neurule is depicted in Fig. 1a. Each condition C_i is assigned a number sf_i, called its *significance factor*. Moreover, each rule itself is assigned a number sf_0, called its *bias factor*. Internally, each neurule is considered as an adaline unit (Fig. 1b). The *inputs* $C_i (i = 1, \ldots, n)$ of the unit are the *conditions* of the rule. The weights of the unit are the significance factors of the neurule and its bias is the bias factor of the neurule. Each input takes a value from the following set of discrete values: [1 (true), 0 (false), 0.5 (unknown)]. This gives the opportunity to easily distinguish between the falsity and the absence of a condition, in

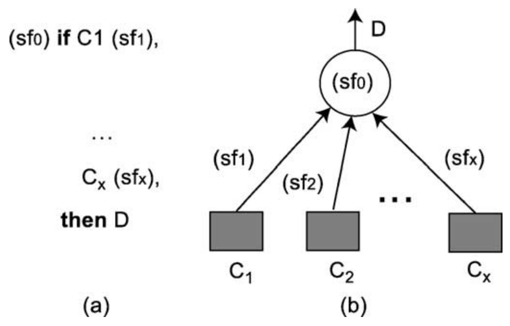

Fig. 1. (a) Form of a neurule and (b) a neurule as an adaline unit.

contrast to symbolic rules. The *output* D, which represents the *conclusion* (decision) of the rule, is calculated via the standard formulas:

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

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

where a is the *activation value* and $f(x)$ the *activation function*, a threshold function. Hence, the output can take one of two values ('-1', '1') representing failure and success of the rule, respectively.

The general syntax of a condition C_i and the conclusion D is:

⟨condition⟩:: = ⟨variable⟩⟨l-predicate⟩⟨value⟩
⟨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. ⟨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.

A variable in a condition can be either an *input variable* or an *intermediate variable* or even an *output variable*, whereas a variable in a conclusion can be either an intermediate or an output variable. An input variable takes values from the user (input data), whereas intermediate or output variables take values through inference since they represent intermediate and final conclusions, respectively. We also distinguish between intermediate and output neurules. An *intermediate neurule* is a neurule having an intermediate variable in its conclusion. An *output neurule* is one having an output variable in its conclusion.

2.2. Neurule base construction

Neurules can be produced either from *symbolic rules*, thus exploiting existing symbolic rule bases, or from *empirical data*. The symbolic rules should have the same syntax as that in Fig. 1a, without the bias and significance factors. In this case, ',' denotes conjunction.

The process of producing neurules from existing symbolic rules is presented in Hatzilygeroudis and Prentzas (2000). According to that process, symbolic rules are organized into *merger sets*. Each merger set contains symbolic rules having the same conclusion. An adaline unit is initially assigned to each merger set. Each unit (rule) is individually trained via the well-known Least Mean Square (LMS) algorithm. Its training set is based on the truth table of the combined logical function of the rules (i.e. the disjunction of the conjunctions of their conditions) in the merger set. Thus, from each merger set, one neurule is produced.

Table 1
Symbolic rules of a merger set

R1 **if** patient-class is human0–20, pain is night, fever is no-fever, ant-reaction is none **then** disease-type is primary-malignant	R2 **if** patient-class is human21–35, pain is night, ant-reaction is none **then** disease-type is primary-malignant
R3 **if** patient-class is human21–35, pain is continuous, fever is no-fever, ant-reaction is none **then** disease-type is primary-malignant	

As an example, Table 1 presents three symbolic rules (R1, R2, R3), from a medical application, that constitute a merger set, that is they have the same conclusion. In Table 2, the neurule (NR1) produced from that merger set is presented. NR1 is said to be a *merger* of R1, R2 and R3. NR1 fires (i.e. produces its conclusion) for the same input data as that for R1–R3. In other words, NR1 substitutes for R1–R3. Notice the potential reduction in the size of the rule-base (from three rules with 11 conditions one neurule with six conditions has been produced).

However, if the patterns in the training set of a neurule form a non-linear set, the symbolic rules of the merger set cannot be merged into one neurule, thus more than one neurule having the same conclusion are produced (see an example in a similar situation below, in the production of neurules from empirical data).

The production of more than one neurule for the same conclusion is a disadvantage of neurules, because in this way the same knowledge is represented more than once. However, the produced neurule base again is significantly smaller than an equivalent of symbolic rules (Section 6).

The process of producing neurules from empirical data is presented in Hatzilygeroudis and Prentzas (2001b). Empirical data is given in the form of knowledge patterns. A *knowledge pattern* relates a number of *variables* with

Table 2
Neurule produced from the merger set of Table 1

NR1 (−7.8) **if** patient-class is human21–35 (6.9), pain is night (6.4), ant-reaction is none (6.3), patient-class is human0–20 (3.0), fever is no-fever (2.7), pain is continuous (2.6) **then** disease-type is primary-malignant

Table 3
Knowledge patterns for variable ‘mark-level’

Solution-attempts	Requested-examples	Assistance-times	Mark-level
1	0	1	High
2	1	1	Average
1	>1	0	High
2	0	1	Average
>2	1	>1	Low
1	>1	>1	Average
>2	0	1	Low
2	1	0	Average
2	>1	>1	Low
>2	0	0	Average

a tuple of (their) values. Each variable can take values from a set of *discrete values*. Each pattern has the following format:

$$[v_1, v_2, \ldots, v_n, d]$$

where d is the value of an intermediate or output variable and $v_i (i = 1, n)$ are the values of the variables it depends on. For example, Table 3 contains 10 knowledge patterns concerning the intermediate variable ‘mark-level’ and the variables it depends on (‘solution-attempts’, ‘requested-examples’, ‘assistance-times’), related to a tutoring system domain.

The main objective of the process is to create one neurule for each discrete value of each intermediate or output variable in the domain. However, due to possible non-linearity of the knowledge patterns, this is not always feasible (as also happens in the case of producing neurules from symbolic rules). So, finally more than one neurule are produced for some value(s).

For example, the neurules produced from the knowledge patterns of Table 3 are presented in Table 4. Notice that, although there is only one neuruele for the ‘low’ and ‘high’ values of ‘mark-level’ (NP1, NP4), there are two neurules for the value ‘average’ (NP2, NP3). This is because the patterns in the corresponding training set for ‘average’ value form a non-linear set.

2.3. Neurule-based inference

The neurule-based inference engine gives pre-eminence to symbolic reasoning, based on a backward chaining strategy. The inference engine uses the *working memory*, which contains *facts* acquired from the user prior to the inference process (i.e. initial input data) and/or produced during the inference process. A fact has the same format as a condition/conclusion of a symbolic rule.

As soon as the initial input data is given and put in the working memory, the output neurules are considered for evaluation. One of them is selected for evaluation. Selection is based on the order they are stored. Evaluation of a neurule is based on neurocomputing measures. A neurule fires if

Table 4
Neurules produced from the knowledge patterns of Table 3

NP1	NP3
(−0.5) if assistance-times is 0 (−4.6),	(−2.6) if solution-attempts is 2 (−6.2),
solution-attempts is 1 (−3.8),	requested-examples is 1 (−6.0),
solution-attempts is >2 (3.0),	assistance-times is 1 (−5.7),
assistance-times is >1 (2.8),	assistance-times is >1 (4.7),
requested-examples is 0 (−2.0),	requested-examples is 0 (3.2),
requested-examples is 1 (0.90),	assistance-times is 0 (−2.7),
requested-examples is >1 (−0.70),	solution-attempts is >2 (2.6),
solution-attempts is 2 (−0.30),	solution-attempts is 1 (−1.0),
assistance-times is 1 (−0.30)	requested-examples is >1 (−0.2)
then mark-level is low	then mark-level is average
NP2	NP4
(−1.0) if assistance-times is >1 (−4.1),	(−0.8) if solution-attempts is 1 (3.2),
solution-attempts is >2 (−2.8),	assistance-times is >1 (−2.7),
requested-examples is >1 (−2.2),	solution-attempts is 2 (−2.6),
assistance-times is 1 (1.6),	requested-examples is 1 (−2.6),
requested-examples is 1 (1.5),	solution-attempts is >2 (−2.5),
solution-attempts is 2 (1.3),	requested-examples is 0 (1.4),
assistance-times is 0 (1.3),	assistance-times is 1 (1.0),
requested-examples is 0 (−0.6),	requested-examples is >1 (−0.7),
solution-attempts is 1 (−0.4)	assistance-times is 0 (−0.3)
then mark-level is average	then mark-level is high

the output of the corresponding adaline unit is computed to be ‘1’ after evaluation of its conditions. A neurule is said to be ‘blocked’ if the output of the corresponding adaline unit is computed to be ‘−1’ after evaluation of its conditions.

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

The conditions of a neurule are organized in descending order of the absolute value of their significance factors. This facilitates inference. Thus, when a neurule is examined in the inference process, not all of its conditions need to be evaluated. To achieve this, we define for each neurule the *known sum* (*ks*) and the *remaining sum* (*rs*) as follows:

$$ks = sf_0 + \sum_{\mathrm{cond}_i \in E} sf_i C_i$$

$$rs = \sum_{\mathrm{cond}_i \in U} |sf_i|$$

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, ks is the sum of the values of the already known (i.e. evaluated) conditions (inputs) of the corresponding neurule and rs represents the largest possible weighted sum of the remaining (i.e. unevaluated) conditions of the neurule.

When a neurule is under evaluation, ks and rs are (re)calculated after evaluation of each condition of the neurule. When $|ks| > rs$, for a certain neurule, then evaluation of its conditions stops, 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 $ks < 0$, or ‘1’, if $ks > 0$. The condition after the evaluation of which the output of the neurule can be deduced is called the *critical condition*.

Also, we introduce the following heuristics that further improve the performance of the neurules. First, if a condition evaluates to ‘true’ (e.g. ‘assistance-times is >1 (−4.1)’ in NP2, Table 4), then the unevaluated conditions containing the same variable, called its *homonymous conditions* (i.e. ‘assistance-times is 1 (1.6)’ and ‘assistance-times is 0 (1.3)’) do not contribute to the remaining sum, because they certainly evaluate to ‘false’, hence they result in a zero value. In any other case, from unevaluated conditions with the same variable, only the one with the maximum absolute factor value contributes to the remaining sum. For example, after evaluation of condition ‘assistance-times is >1 (2.8)’ in NP1 (Table 4), only condition ‘requested-examples is 0 (−2.0)’ contributes to the remaining sum, because its homonymous conditions, ‘requested-examples is 1 (0.90)’ and ‘requested-examples is >1 (−0.70)’, have less absolute factor values and these three conditions cannot be simultaneously true. Also, the last two conditions do not contribute, because they are certainly false (one of their homonymous conditions has already evaluated to ‘true’).

To illustrate how inference is made, we present the following example. Suppose that we have a knowledge base containing the four neurules of Table 4 (Section 2.2) and a working memory with the following facts in it: {‘solution-attempts is 1’, ‘requested-examples is >1’, ‘assistance-times is >1’}. The inference engine starts examining each one of the neurules, to find whether any of them fires (succeeds). As mentioned, the neurules are evaluated in the order they are stored. The known and remaining sums are calculated for each neurule after each condition evaluation (recall that the value of a condition is ‘1’ if it is true and ‘0’ if it is false). The ks and rs for each neurule after the critical condition (specified within the parentheses) evaluation are as follows:

NP1 (after evaluation of its fifth condition): $ks_1 = -1.5$, $rs_1 = 0.9$ (failure, because $|ks_1| = 1.5 > rs_1$ and $ks_1 < 0$);
NP2 (after evaluation of its third condition): $ks_2 = -7.3$, $rs_2 = 1.6$ (failure, because $|ks_2| = 7.3 > rs_2$ and $ks_2 < 0$);
NP3 (after evaluation of its seventh condition): $ks_3 = 2.1$, $rs_3 = 1.2$ (success, because $|ks_3| = 2.1 > rs_3$ and $ks_3 > 0$).

Because NP3 succeeds, the inference process stops and the conclusion ‘mark-level is average’ is produced and put in the working memory. Notice that not all of the conditions of the neurules have been evaluated to reach a conclusion.

Experimental results have shown that neurule-based inference is more efficient than the corresponding symbolic rule-based inference. The main reason for this is the fact that a neurule is a merger of usually more than one symbolic rule having the same conclusion and thus the total number of rules and the conditions in the rule base is reduced. In this way, the number of the rules and the conditions participating in the inference process is reduced. This is intensified due to the heuristics introduced above (Section 6).

Another advantage of neurule-based reasoning compared to symbolic rule-based reasoning is the ability to reach conclusions from neurules even if some of the conditions are unknown (i.e. from partial input). This is not possible in symbolic rule-based reasoning. A symbolic rule needs all of its conditions to be known in order to produce a conclusion.

For example, if ‘solution-attempts is unknown’ is in the working memory (instead of ‘solution-attempts is 1’) in the above example, then again NP3 succeeds, whereas NP1 and NP2 fail. The policy we follow in the case of a variable with ‘unknown’ as its value is as follows: the contribution to the known sum is the average value of the significance factors of all the homonymous conditions of that variable multiplied by 0.5. In our case, the average value of the significance factors of the homonymous conditions of ‘solution-attempts’ is $(-6.2 + 2.6 - 1.0)/3 = -1.53$. Also, after that contribution, which happens when one of the homonymous conditions is met, the rest of them do not contribute to either the known or the remaining sum any more. To illustrate the evaluation process, we present evaluation of NP3 in some detail:

Condition	*ks*	*rs*
1	$-2.6 + (-1.53) = -4.13$	$6.0 + 5.7 = 11.7$
2, 3	$-4.13 + 0 = -4.13$	$4.7 + 3.2 = 7.9$
4	$-4.13 + 4.7 = 0.57$	3.2
5	$0.57 + 0 = 0.57$	0.2

Because, after the evaluation of the fifth condition (the critical condition), $|ks| = 0.57 > rs = 0.2$ and $ks > 0$, NP3 succeeds and its conclusion is added to the working memory. This does not happen with symbolic rules, where evaluation stops in such cases.

2.4. Neurule-based explanation mechanism

As it is well known, one of the advantages of symbolic rule-based reasoning is the ease of producing explanations for the conclusions. This is achieved by tracing the fired rules in the reverse order.

The neurule-based explanation mechanism justifies inferences by producing a set of symbolic (if–then) rules, thus providing an explanation of how the conclusions were reached from a neurule base. To this end, for each of the fired output neurules, the explanation mechanism generates an if–then rule whose conclusion is the neurule's conclusion and its conditions are the *necessary conditions* of the neurule. The necessary conditions of a fired neurule are (a) those evaluated to 'true' and (b) the first of the remaining (unevaluated) conditions that evaluates to 'true'. The need for (b) comes from experimental results. We noticed that conditions that evaluated to 'true' were not adequate by themselves to produce explanation rules. There was always one of the remaining conditions missing. This is basically due to deficiencies of the training algorithm (LMS). In addition, for each condition containing an intermediate variable, an if–then rule is produced based on an evaluated (fired) neurule having that condition as its conclusion, in the same way. This process recurses.

The above explanation process is different from that in Hatzilygeroudis and Prentzas (2001a), because a different set of condition values is used here, i.e. ('1', '0', '0.5') is used instead of ('1', '-1', '0'), for naturalness reasons. This change affects the explanation mechanism.

So, the variables in the conditions of the explanation rules are either input variables whose values were given by the user or intermediate variables whose values were derived during inference. The conclusions of the explanation rules correspond to the derived final conclusions (i.e. the ones containing output variables) and to the intermediate conclusions that contributed to drawing the final conclusions.

For example, in the initial above example, because there is only one final conclusion and not any intermediate ones, only one explanation rule is produced, derived from NP3:

```
EXR1
if   assistance-times is >1,
     solution-attempts is >1
then mark-level is average
```

According to the explanation mechanism, the necessary conditions of EXR1 are (a) the conditions that evaluated to 'true' ('assistance-times is >1') and (b) the first of the unevaluated conditions that evaluates to 'true' ('solution-attempts is >1'), given the content of the working memory at that stage.

3. The hybrid architecture

In Fig. 2, the architecture of the hybrid system implementing the method for integrating neurule- and case-based reasoning is presented. The run-time system (in the dashed shape) consists of the following modules: the *working memory*, the *hybrid inference mechanism*, the *hybrid explanation mechanism*, the *neurule base* and the *indexed case library*.

The neurule base contains neurules that represent general domain knowledge. The neurules can be produced by conversion from an existing *symbolic rule base*, as specified in Section 2.2. This process is performed by the *rules to neurules* module. Alternatively, neurules can be produced from existing *empirical data*, as specified in Section 2.2 too. This process is performed by the *patterns to neurules* module.

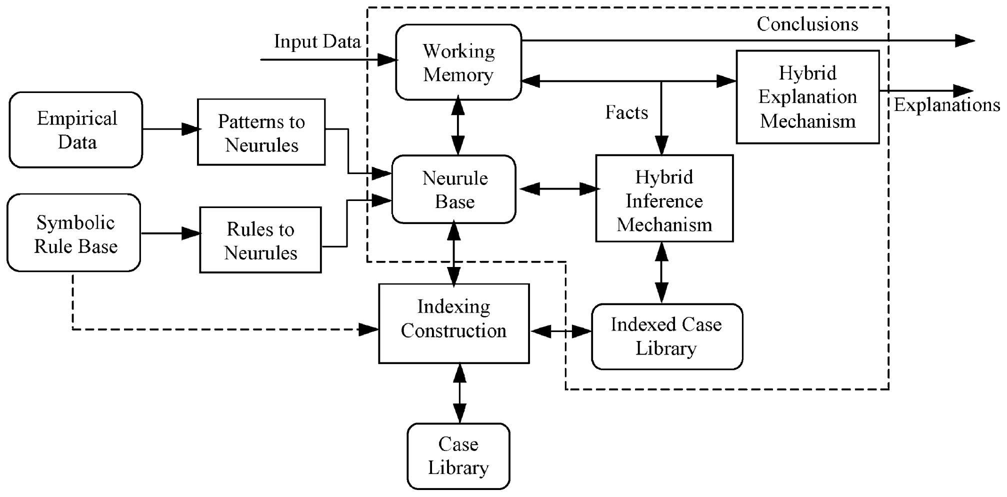

Fig. 2. The hybrid architecture.

Table 5
Example neurule

NR2
(− 13.1) **if** pain is continuous (6.9),
fever is high (5.2),
fever is medium (4.8),
patient-class is human21–35 (2.7),
patient-class is human0–20 (1.6)
then disease-type is inflammation

The neurule base is used to index a *case library*. Thus, the *indexed case library* is derived. The cases in the indexed case library act as exceptions to the neurules in the neurule base. Each case is formalized as a set of attribute–value pairs. In a case, we distinguish between its *input attributes* and its *output attribute*. The output attribute is the one that represents the (observed) outcome/decision produced/made in a case and matches the variable of the conclusion of the associated neurule. The rest attributes of a case are its input attributes. So, a case is represented as a tuple of the following format:

$$[i_1, i_2, \ldots, i_m, o]$$

where $i_j (j = 1, m)$ are values of the *input attributes* and o is the value of the *output attribute* of the case. The process of establishing an indexing between neurules and cases is performed offline by the *indexing construction* module and is presented in Section 4.

The *hybrid inference mechanism* makes inferences by combining neurule- and case-based reasoning. It takes into account the facts contained in the working memory, the neurules in the neurule base and the cases in the indexed case library. The hybrid inference mechanism is presented in Section 5.

The *hybrid explanation mechanism* provides explanations in the way described in Section 2.4.

4. Indexing

Indexing concerns organizing the available cases so, that the combined neurule- and case-based reasoning can be performed. The neurules contained in the neurule base are used to index cases representing exceptions to them. A case constitutes an exception to a neurule if its attribute values satisfy a sufficient number of conditions of the neurule (so that it can fire), but the neurule's conclusion contradicts the corresponding attribute value of the case. Cases that are exceptions to neurules are considered of great importance, as they fill gaps in the knowledge represented by the neurules. During inference, exceptions may assist in reaching the right conclusion.

We distinguish two indexing processes: *direct indexing* and *indirect indexing*. Direct indexing concerns associating available cases to neurules, which is the normal case. Indirect indexing deals with associating already indexed cases, by existing symbolic rules, to neurules produced from the symbolic rules. The type of the available knowledge determines which process should be used.

4.1. Direct indexing

The objective of the direct indexing process is to associate the available neurules with the available cases, which will constitute their exceptions. The direct process is as follows:
For each case,
Until all its input attribute values have been considered,

1. Perform neurule-based reasoning taking as initial data the input attribute values of the case.
2. If a neurule fires, check whether its conclusion value matches the corresponding output attribute value of the case, given that the variable of the conclusion is the same as the output attribute. If it does not, mark the case as an exception to this neurule.

As an example, to demonstrate how the indexing process works, we use neurule NR2 (produced in a similar Way to NR1), presented in Table 5, and the two example cases in Table 6. Only the most important attributes of the cases are shown in Table 6. The cases, however, also posses other attributes (not shown in Table 6).

'disease-type' is the output attribute that corresponds to the neurule's conclusion variable. Let ks_1 and rs_1 be the known and remaining sum of the neurule, respectively, for the first case. Also, let ks_2 and rs_2 be the known and remaining sum, respectively, for the second case. Evaluation of conditions for the first (second) case continues until $|ks_1| > |rs_1| (|ks_2| > |rs_2|)$. Recall that when a condition evaluates to 'true' it gets the value '1', whereas in case it is false the value '0'. The tracings of the evaluations of neurule NR2 for the two cases of Table 6 are presented in Table 7. Notice that the final values of the known and remaining sums are:

$$ks_1 = 1.3 > 0, \qquad rs_1 = 1.6$$

$$ks_2 = 0.6 > 0, \qquad rs_2 = 0.$$

Table 6
Example cases

Case no.	*Patient-class*	*Pain*	*Fever*	*Ant-reaction*	*Joints-pain*	*Disease-type*
1	human21–35	continuous	medium	high	yes	special-arthritis
2	human0–20	continuous	high	high	no	inflammation

Table 7
NR2 evaluation tracings for the cases of Table 4

Condition no.	Case 1		Case 2	
	ks_1	rs_1	ks_2	rs_2
1	$-13.1+6.9=-6.2$	$5.2+4.8+2.7+1.6=14.3$	$-13.1+6.9=-6.2$	$5.2+4.8+2.7+1.6=14.3$
2	$-6.2+0=-6.2$	$4.8+2.7+1.6=9.1$	$-6.2+5.2=-1.0$	$4.8+2.7+1.6=9.1$
3	$-6.2+4.8=-1.4$	$2.7+1.6=4.3$	$-1.0+0=-1.0$	$2.7+1.6=4.3$
4	$-1.4+2.7=1.3$	1.6	$-1.0+0=-1.0$	1.6
5			$-1.0+1.6=0.6$	0

So, the input attribute values of both cases give a positive known sum for the neurule's conditions. The fact that the known sums are positive means that both cases concern the 'inflammation' disease-type conclusion. However, only the output value of the disease-type of the second case matches the neurule's conclusion. The corresponding output value of the first case contradicts the conclusion, hence that case is indexed as an exception to NR2.

4.2. Indirect indexing

Indirect indexing is used when existing symbolic rules with associated exception cases are available, for example like those in Golding and Rosenbloom (1996). The process is as follows:

1. Convert the symbolic rules into neurules via the 'rules to neurules' module
2. Associate the produced neurules with the exception cases of the symbolic rules belonging to their merger sets.

Of course, existing symbolic rules are converted to the format described in Section 2.2, before the above process is applied.

As an example, consider the three symbolic rules (R1, R2, R3) presented in Table 1. Table 8 presents some of their exception cases. The symbolic rule, to which each of these example cases is an exception, is shown in the 'disease-type' column of the table in parentheses. As mentioned in Section 2.2, by merging those three symbolic rules neurule NR1 is produced (Table 2). The cases in Table 8 are now indexed as exceptions to NR1. The produced neurule has now four exception cases.

Notice that the exceptions of NR1 are more than the average number of exceptions that the symbolic rules in its merger set had. So, a potential disadvantage of this process could be the fact that in average a produced neurule will be associated with more exception cases than each of its merged symbolic rules. This may negatively affect the case-based reasoning part of the inference process. However, the explanation rule produced from the neurule can be used to surpass this deficiency by limiting the exception cases considered by case-based reasoning (Section 5).

5. The hybrid inference mechanism

The hybrid inference mechanism combines neurule-based reasoning with case-based reasoning. The combined inference process mainly focuses on the neurules. The exception cases are considered only when sufficient conditions of a neurule are fulfilled so that it can fire. Then, firing of the neurule is suspended and case-based reasoning is performed on its exception cases. To reduce the number of exception cases to be taken into account, the corresponding explanation rule is produced. Then, only the exception cases that match the explanation rule are considered. Matching the explanation rule means making all its conditions true. The results produced by case-based reasoning are evaluated in order to assess whether the neurule will fire or whether the conclusion proposed by the exception case will be considered valid.

Case-based reasoning and evaluation of its results is performed as in Golding and Rosenbloom (1996). According to that, case-based reasoning tries to find similarities between input data and exception cases. If such a similarity is found, an analogy rule encompassing the similar attribute values between the indexed case and the input case is produced. The analogy rule is then evaluated. Evaluation is

Table 8
Exception cases indexed by the symbolic rules in Table 3

Case no.	Patient-class	Pain	Fever	Ant-reaction	Joints-pain	Disease-type
1	human0–20	night	no-fever	none	yes	inflammation (R1)
2	human21–35	night	no-fever	none	no	arthritis (R2)
3	human21–35	continuous	no-fever	none	no	primary-benign (R3)
4	human21–35	continuous	no-fever	none	yes	chronic-inflammation (R3)

based on two factors: the *similarity degree* between the input case and the exception case and how well the analogy rule works (generalizes) on other exception cases. If the analogy between the input case and an exception case is proved to be compelling, the conclusion supported by the case is considered valid and the associated neurule becomes 'blocked'. Compellingness is precisely defined in Golding and Rosenbloom (1996) and is not of concern in this paper.

The basic steps of the inference process combining neurule- and case-based reasoning are as follows:

1. Perform neurule-based reasoning on the input data.
2. If an adequate number of the conditions of a neurule are fulfilled (so that it can fire), then
   2.1. If the neurule has no associated exception cases, activate firing and insert its conclusion into the working memory.
   2.2. If the neurule is associated with exception cases, suspend firing. Produce the corresponding explanation rule and perform case-based reasoning on the neurule's associated exception cases that match the explanation rule.
   2.3. If the analogy rule proposed by case-based reasoning is compelling, insert the conclusion supported by the exception case into the working memory and mark the neurule as 'blocked'. Otherwise, mark the neurule as 'fired' and insert its conclusion into the working memory.

We now present a simple example. For the sake of simplicity, we consider that only the similarity degree between the input data and the exception case is taken into account for its evaluation. The similarity degree is defined as the number of the common attribute–value pairs of the two cases (the input and the exception). Also, we consider that the exception case is valid if the similarity degree is equal to or greater than the the number of conditions of the produced explanation rule.

Suppose that the case in Table 9 is given as input to neurule NR1 (Table 2).

Evaluation of the rule stops after its third condition (critical condition) has been evaluated, because the absolute value of the known sum ($-7.8 + 6.9 + 6.4 + 6.3 = 11.8$) is greater than the value of the remaining sum ($2.7 + 2.6 = 5.3$).

The explanation rule (EXR2) produced from neurule NR1, via the explanation mechanism (Section 2.4), is shown in Table 10. Firing the neurule is suspended and

Table 9
An input case to neurule NR1

Patient-class	Pain	Fever	Ant-reaction	Joints-pain
human21–35	night	high	none	no

Table 10
Explanation rule produced for neurule N1

```
EXR2
if     patient-class is human21-35,
       pain is night,
       ant-reaction is none
then disease-type is primary-malignant
```

the exception cases matching the explanation rule are considered in the case-based reasoning process. From the exception cases shown in Table 8, only the second one matches the explanation rule. So, only this exception case is considered. Notice that EXR2 is the same as R2 (Table 1) and that the second case of Table 8 was associated to R2. So, it is like extracting from the neurule the symbolic rule from the neurule's merger set, which would fire.

The similarity degree between the input and the exception case is 3, which is equal to the number of the conditions of the explanation rule. So, the exception case is considered valid and its conclusion, based on its output attribute–value pair ('disease-type is arthritis') is produced and put in the working memory. NR1 is blocked.

Suppose now that the same input case was given to the three symbolic rules shown in Table 1. In the symbolic rule-based reasoning part of the inference, the order that the rules will be considered is R1, R2, and R3. The conditions of rule R1 are not satisfied. However, all conditions of rule R2 are satisfied. Thus, four rule conditions should be examined (one for R1 and three for R2), instead of the three conditions examined in neurule-based reasoning. Firing of rule R2 is suspended and its indexed exception cases (i.e. the second case in Table 8) will be considered by case-based reasoning.

So, both types of inference mechanisms produce the same results. However, the inference mechanism combining neurule- and case-based reasoning requires the examination of fewer conditions (Section 6).

6. Experimental results

To test the effectiveness of our approach, we used a symbolic rule base concerning a medical application domain (Hatzilygeroudis, Vassilakos, & Tsakalidis, 1997) The symbolic rule base contained 58 symbolic rules acquired by interviewing an expert. The symbolic rules were indexing available exception cases in order to improve their accuracy. This combined symbolic rule base and indexed case library will be referred to as SRCL. By using the rules-to-neurules module, the symbolic rules were converted to neurules. In total, 34 neurules were produced (i.e. a 41% in rule number reduction and about 30% in condition number reduction). The exception cases of the merged symbolic rules were indexed by the produced neurules, as specified in Section 4. The combined neurule base and indexed case library will be referred to as NRCL.

Inferences were run for both SRCL and NRCL. Inferences from SRCL were performed using the inference mechanism combining rule- and case-based reasoning as described in Golding and Rosenbloom (1996). Inferences from NRCL were performed according to the inference mechanism integrating neurule- and case-based reasoning, introduced here. As expected, inferences produced the same conclusions from both SRCL and NRCL for the same variable-value data. However, inferences from NRCL required evaluation of fewer conditions than the corresponding inferences from SRCL.

Table 11 presents experimental results regarding inferences from SRCL and NRCL. It presents results regarding the number of visited rules as well as the number of evaluated conditions. The table also presents whether the conclusion was derived as an exception or not (column 'Exception Occurred').

As can be seen from the table, there is an average 45% reduction in the rules visited in NRCL. Furthermore, about 25% fewer conditions were evaluated in inferences from NRCL. Finally, the integration with case-based reasoning improved the accuracy of the inference mechanism in about 30% of the inferences.

Table 11
Experimental results

Inference no.	Rules visited, SRCL/NRCL	Conditions evaluated, SRCL/NRCL	Exception occurred
1	8/4	17/10	No
2	11/6	26/19	Yes
3	13/8	26/22	No
4	17/10	34/27	No
5	20/10	28/19	No
6	21/10	32/24	No
7	22/10	31/23	No
8	23/10	35/22	No
9	24/11	43/31	No
10	25/12	47/37	No
11	26/13	48/39	Yes
12	29/15	57/44	Yes
13	30/15	59/46	Yes
14	31/15	63/42	No
15	32/16	58/43	Yes
16	34/18	70/48	No
17	35/19	64/49	Yes
18	41/23	77/57	No
19	42/24	77/66	No
20	43/24	78/63	Yes
21	44/24	78/58	No
22	47/26	86/71	No
23	48/27	86/66	No
24	49/27	90/64	Yes
25	51/28	76/60	No
26	54/30	72/57	No
Total	820/435	1458/1107	

7. Related work

According to our knowledge, there are no systems integrating rules, cases and neural networks. Our approach integrates an hybrid representation formalism, namely neurules, where the rule-based approach dominates, with a case-based approach. Thus, we present here, as related work, approaches that integrate rule- and case-based reasoning. In doing that we also propose a new high-level categorization scheme.

Various methods integrating rule-based with case-based reasoning have been developed during the last years. Legal reasoning seems to be a popular application field for the integrated approaches. This is so, because legal reasoning concerns rules containing terms (i.e. open-textured terms) which are not well defined and need an integration with cases to reach (or enhance) a conclusion. Examples of integrated approaches used in legal reasoning include GREBE (Branting, 1991, 1999), CABARET (Rissland & Skalak, 1991), IKBALS II (Vossos, Zeleznikow, Dillon, & Vossos, 1991; Zeleznikow, Vossos, & Hunter, 1994), SHYSTER-MYCIN (O'Callaghan, Popple, & McCreath, 2003). Other application fields of the integrated approaches include medicine (Bellazi, Montani, Portinale, & Riva, 1999; Bichindaritz & Sullivan, 1998a,b; Park, Oh, Jeong, & Park, 2000), surname pronunciation, ANAPRON (Golding & Rosenbloom, 1996), part-of-speech tagging (Lopes & Jorge, 2000a,b), music (Sabater, Arcos, & Lopez de Mantaras, 1998), design of nutrition menus (Marling et al., 1999).

In Golding and Rosenbloom (1996), the approaches integrating rule- and case-based reasoning are distinguished into two basic categories: *efficiency-improving* and *accuracy-improving methods*. The former concern integration methods in which rules and cases are dependent, meaning that one representation scheme was derived from the other (i.e. rules derived from cases or vice versa), and the efficiency of the integrated scheme exceeds the efficiency that could have been achieved with rules or cases alone (Koton, 1988; Veloso, 1992). The latter involves approaches in which the two representation schemes are independent and their integration results in improved accuracy compared to each component representation scheme working individually (Branting, 1991; Golding & Rosenbloom, 1996; Rissland & Skalak, 1991).

The above categorization scheme may not be always adequate to classify approaches integrating rule- and case-based reasoning, because it bases its high-level classification on combined criteria. It supposes that (a) efficiency improvement goes with systems where cases and rules are dependent and (b) accuracy improvement goes with systems where cases and rules are independent. However, there may be systems that improve efficiency, where rules and cases are not dependent or vice versa. For example in (Bellazi et al., 1999), cases and rules are independent and mainly efficiency is improved.

We propose a high-level categorization based on the way that rules and cases are integrated. So, we distinguish three main categories: *rule-dominant*, *case-dominant* and *balanced* approaches, depending on which of the two component approaches dominates.

The rule-dominant category includes approaches like the ones used in (Dutta & Bonissone, 1993; Golding & Rosenbloom, 1996; Surma & Vanhoof, 1995, 1998), IKBALLS II (Lopes & Jorge, 2000a,b; Park et al., 2000; Vossos et al., 1991; Zeleznikow, Vossos, & Hunter, 1994) and SHYSTER-MYCIN (O'Callaghan et al., 2003). In Dutta and Bonissone (1993), rules are used to represent domain knowledge in a system called MARS, which is used to make decision about mergers and acquisitions in the corporate domain. Case-based reasoning is activated by selective rules during the course of rule-based reasoning. Golding and Rosenbloom (1996) propose a general architecture for integrating rule-based with case-based reasoning, where the reasoning process is mainly guided by rules and cases are used before final conclusions are to be produced. Experimental results based on ANAPRON, an implementation of the architecture, have demonstrated the effectiveness of the integration. In the approach described in Surma and Vanhoof (1995, 1998), the knowledge base contains rules representing standard situations and cases representing exceptions or non-standard situations. The contents of the knowledge base are produced from an initial case base, whose cases are split into two types: standard cases, used to induce the rules, and exception cases. The splitting heuristics play an important role in the accuracy of the approach. IKBALS II is an approach applied to a legal application domain. Cases correspond to exceptional situations and are used when rules run out or prove insufficient in drawing conclusions. The approach described in Lopes and Jorge (2000a,b) is used for part-of-speech-tagging (i.e. morpho-syntactic disambiguation). Rules are induced from examples in an iterative procedure with degrading quality. When the quality of induced rules falls below a threshold, a case-based approach is used with the examples not covered by the induced rules. The case-based reasoning module employs sets of explanations produced from cases through a learning process. In Park et al. (2000) cases play a supplementary role in the reasoning process as they deal with details and exceptions to the rules. Rules handle also uncertainty by adopting the reliability value, a simplified form of certainty factor. In SHYSTER-MYCIN (O'Callaghan et al., 2003), a case-based reasoner, called SHYSTER, is combined with a modified version of the well-known MYCIN expert system. Reasoning focuses mainly on the MYCIN part, calling SHYSTER whenever an 'open-textured' term is encountered.

The balanced approaches category includes approaches like those used in GREBE (Branting, 1991, 1999), CABARET (Rissland & Skalak, 1991), DIAL (Bellazi et al., 1999; Bichindaritz & Sullivan, 1998a,b; Leake, Kinley, & Wilson, 1996). GREBE achieves a coherent interleave of rules and cases in order to effectively handle legal reasoning. At any level of the inference process, it can invoke the rules or the cases of the system. CABARET is also an approach dealing with legal reasoning. The rule-based component and the case-based component act as co-reasoners. A controller observes the operation of the whole system and each co-reasoner separately and decides on how they will proceed in the reasoning process as a whole and individually. DIAL (Leake, Kinley, & Wilson, 1996) is a multi-modal case-based system that incorporates case-based and rule-based components to facilitate tasks such as adaptation and similarity assessment. Internally, rule- and case-based reasoning fall back on each other whenever this is necessary. The approach described in Bichindaritz & Sullivan (1998a,b) has been applied to post-transplant patient care. The inference mechanism integrates closely rules and cases since pattern matching and case-based retrieval is performed in parallel and the conflict set may simultaneously contain rules as well as cases. The approach described in Bellazi et al. (1999) has been applied to diabetic patient management. The innovative aspect is the use of cases in order to dynamically adapt (refine) general parameters of certain rules and thus improve handling of a new situations.

Finally, the case-dominant category includes approaches like those in GYMEL (Sabater et al., 1998) and CAMPER (Marling et al., 1999). GYMEL is a system for harmonizing melodies. In the integration scheme, cases play a more important role than rules since rules are invoked when the cases cannot produce a solution. In this way, a potentially inadequate set of cases is assisted by the general knowledge of rules. CAMPER is a nutritional menu planner built by combining the best features of independent case-based reasoning and rule-based reasoning menu planners, CAMP and PRISM, respectively. In CAMPER, the case-based reasoning component constructs acceptable menus satisfying multiple nutrition constraints. The rule-based component from its part can enhance the proposed menu.

Our approach can be considered as belonging to the rule-dominant category. Notice, however, that hybrid rules are used instead of single rules.

8. Conclusions

In this paper, we present an approach that integrates symbolic rules, neural networks and cases. To achieve it, we integrate a kind of hybrid rules, called neurules, with cases. Neurules integrate symbolic rules with the Adaline neural unit. In the integration, neurules are used to index cases representing their exceptions.

In this way, the accuracy of the neurules is improved. On the other hand, due to neurule-based efficient inference mechanism, conclusions can be reached more efficiently. In addition, neurule-based inference can be performed even if some of the inputs are unknown. This is not possible in symbolic rule-based reasoning. Furthermore, an existing symbolic rule-base with indexed exception cases can be converted into a neurule-base with corresponding indexed exception cases. Finally, empirical data can be used as a knowledge source, which facilitates knowledge acquisition.

Of course, this is not the only way that rules, neural networks and cases can be combined. In this paper, the connectionist approach is integrated in the rules part. Another option would be to integrate a connectionist approach in the cases part. For example, a neural network could be used to assess the similarity between the input and the exception cases, which is an issue for further research.

Most hybrid intelligent systems implemented in the past usually integrate two intelligent technologies, e.g. neural networks and rules, neural networks and fuzzy logic, genetic algorithms and neural networks, etc. (Medsker, 1995). We believe that a new development that should receive interest in the future is the integration of more than two intelligent technologies that can facilitate the solution of complex problems and exploit multiple types of available data sources.

Also, we present in this paper a new high-level categorization of the approaches integrating rule-based and case-based reasoning. Categorization in lower levels is an issue for further research.

Acknowledgements

This work was supported by the Research Committee of the University of Patras, Program Karatheodoris, project No. 2788.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations and system approaches. *Artificial Intelligence Communications*, *7*(1), 39–59.

Aha, D., & Daniels, J. J. (Eds.), (1998). *Case-based reasoning integrations: Papers from the AAAI workshop. Technical Report WS-98-15*, Menlo Park, CA: AAAI Press.

Bellazi, R., Montani, S., Portinale, L., & Riva, A (1999). Integrating rule-based and case-based decision making in diabetic patient management. *Proceedings of the third international conference on case-based reasoning. Lecture notes in computer science. Vol. 1650* (pp. 386–400). Berlin: Springer.

Bichindaritz, I., & Sullivan, K. M. (1998a). In Freuder (Ed.), *Reasoning from knowledge supported by more or less evidence in a computerized decision support system for bone-marrow post-transplant care* (pp. 85–90).

Bichindaritz, I., & Sullivan, K. M (1998). Case-based reasoning in CARE-PARTNER: Gathering evidence for evidence-based medical practice. *Proceedings of the fourth european conference on case-based reasoning. Lecture notes in computer science. Vol. 1488* (pp. 334–345).

Branting, L. K. (1991). Building explanations from rules and structured cases. *International Journal of Man-Machine Studies*, *34*(6), 797–837.

Branting, L. K. (1999). Reasoning with rules and precedents. Dordrecht: Kluwer Academic Publishers.

Cercone, N., An, A., & Chan, C. (1999). Rule-induction and case-based reasoning: hybrid architectures appear advantageous. *IEEE Transactions on Knowledge and Data Engineering*, *11*(1), 164–174.

Dutta, S., & Bonissone, P. P. (1993). Integrating case based and rule based reasoning. *International Journal of Approximate Reasoning*, *8*(3), 163–203.

Freuder (Ed.) (1998). *Multimodal reasoning: Papers from the 1998 AAAI spring symposium.* Technical Report SS-98-04. Menlo Park, CA: AAAI Press.

Fu, L.-M., & Fu, L.-C.:. (1990). Mapping rule-based systems into neural architecture. *Knowledge-Based Systems*, *3*, 48–56.

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. Cambridge, MA: MIT Press.

Ghalwash, A. Z. (1998). A recency inference engine for connectionist knowledge bases. *Applied Intelligence*, *9*(3), 201–215.

Golding, A. R., & Rosenbloom, P. S. (1996). Improving accuracy by combining rule-based and case-based reasoning. *Artificial Intelligence*, *87*(1/2), 215–254.

Gonzalez, A. J., & Dankel, D. D. (1993). The engineering of knowledge-based systems, theory and practice. Englewood Cliffs, NJ: Prentice Hall.

Hatzilygeroudis, I., & Prentzas, J. (2000). Neurules: improving the performance of symbolic rules. *International Journal on AI Tools*, *9*(1), 113–130.

Hatzilygeroudis, I., & Prentzas, J (2001). An efficient hybrid rule-based inference engine with explanation capability. *Proceedings of the 14th international FLAIRS conference* (pp. 227–231). Menlo Park, CA: AAAI Press.

Hatzilygeroudis, I., & Prentzas, J. (2001b). Constructing modular hybrid knowledge bases for expert systems. *International Journal on Artificial Intelligence Tools*, *10*, 87–105.

Hatzilygeroudis, I., Vassilakos, P. J., & Tsakalidis, A. (1997). XBONE: A hybrid expert system supporting diagnosis of bone diseases. In C. Pappas, N. Maglaveras, & J.-R. Scherrer (Eds.), *Proceedings of the medical informatics Europe'97 (MIE'97)* (pp. 259–299). Amsterdam: IOS Press.

Kolodner, J. (1993). Case-based reasoning. San Mateo, CA: Morgan Kaufmann Publishers.

Koton, P. (1988). *Reasoning about evidence in causal explanations. Proceedings of the AAAI-88*, Menlo Park, CA: AAAI Press.

Leake, D. B (1995). Combining rules and cases to learn case adaptation. *Proceedings of the 17th annual conference of the cognitive science society*.

Leake, D. B. (Ed.), (1996). *Case-based reasoning: Experiences, lessons and future directions*. Menlo Park, CA: AAAI Press/MIT Press.

Leake, D. B., Kinley, A., & Wilson, D (1996). Acquiring case adaptation knowledge: a hybrid approach. *Proceedings of the 13th national conference on artificial intelligence* (pp. 684–689). Mento Park, CA: AAAI Press.

Lopes, A. A., & Jorge, A (2000). Combining rule-based and case-based learning for iterative part-of-speech tagging. *Proceedings of the fifth european conference on case-based reasoning. Lecture notes in computer science. Vol. 1898* (pp. 26–36). Berlin: Springer.

Lopes, A. A., & Jorge, A (2000). Integrating rules and cases in learning via case explanation and paradigm shift. *Proceedings of the international joint ibero-american conference on artificial intelligence and brazilian symposium on artificial intelligence (IBERAMIA-SBIA). Lecture notes in computer science. Vol. 1952* (pp. 33–42). Berlin: Springer.

Marling, C. R., Petot, G. J., & Sterling, L. S. (1999). Integrating case-based and rule-based reasoning to meet multiple design constraints. *Computational Intelligence*, *15*(3), 308–332.

Medsker, L. R. (1995). Hybrid intelligent systems. Dordrecht: Kluwer Academic Publishers.

O' Callaghan, Th. A., Popple, J., & McCreath, E (2003). SHYSTER-MYCIN: a hybrid legal expert system. *Proceedings of the ninth international conference on AI and law* (pp. 103–104).

Park, H.-J., Oh, J.-S., Jeong, D.-U., & Park, K.-S. (2000). Automated sleep stage scoring using hybrid rule- and case-based reasoning. *Computers and Biomedical Research*, *33*, 330–349.

Prentzas, J., & Hatzilygeroudis, I (2002). Integrating hybrid rule-based with case-based reasoning. *Proceedings of the ECCBR-2002, Aberdeen, Scotland, UK* (September 2002) as Craw, S., & Preece, A. *Advances in case-based reasoning, LNAI 2416* (pp. 336–349). Berlin: Springer.

Rissland, E. L., & Skalak, D. B. (1991). CABARET: rule interpretation in a hybrid architecture. *International Journal of Man–Machine Studies*, *34*(6), 839–887.

Sabater, J., Arcos, J. L., & Lopez de Mantaras, R. (1998). In Freuder (Ed.), *Using rules to support case-based reasoning for harmonizing melodies* (pp. 147–151).

Surma, J., & Vanhoof, K (1995). Integrating rules and cases for the classification task. *Proceedings of the first international conference on case-based reasoning. Lecture notes in computer science. Vol. 1010* (pp. 325–334). Berlin: Springer.

Surma, J., & Vanhoof, K. (1998). In Freuder (Ed.), *An empirical study on combining instance-based and rule-based classifiers* (pp. 130–134).

Towell, G., & Shavlik, J. (1993). Extracting refined rules from knowledge-based neural networks. *Machine Learning*, *13*(1), 71–101.

Towell, G., & Shavlik, J. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, *70*(1/2), 119–165.

Veloso, M. M (1992). *Learning by analogical reasoning in general problem solving*. PhD Thesis. Technical Report CMU-CS-92-174. Carnegie Mellon University, Pittsburg, PA.

Vossos, G., Zeleznikow, J. Dillon, Th., & Vossos, V (1991). An example of integrating legal case-based reasoning with object-oriented rule-based systems: IKBALS II. *Proceedings of the third international conference on artificial intelligence and law* (pp. 31–41).

Zeleznikow, J., Vossos, G., & Hunter, D. (1994). The IKBALS Project: multimodal reasoning in legal knowledge based systems. *Artificial Intelligence and Law*, *2*(3), 169–203.