Combining Case-Based Reasoning with Neural Networks in Diagnostic Systems

Eliseo Berni Reategui

A dissertation submitted in partial fulfilment

of the requirement for the degree of

Doctor of Philosophy

of the

University of London

Department of Computer Science

University College London

1997

Abstract

This thesis presents a new approach for integrating Case-Based Reasoning (CBR) with a Neural Network (NN) in diagnostic systems. In the hybrid NN-CBR approach, the neural network makes hypotheses and provides remindings that are used in guiding the search for similar experiences in a library of previous cases. CBR is responsible for the selection of a most similar match for a given problem, as to support a particular hypothesis made by the neural network, or to decide among hypotheses. Items called *diagnosis descriptors* have been created in order to represent in an intelligible way the knowledge stored in the neural network. These descriptors are used for consultation purposes, for confirming or refuting a final result, and for building explanations.

The NN-CBR architecture has been used in the development of a system for the diagnosis of Congenital Heart Diseases (CHD). The system has been evaluated using two cardiological databases with a total of 214 CHD cases. The cases of the first database were collected with the supervision of an expert in CHD, while the cases of the second database were collected without expert supervision. The hybrid system has shown a good performance when diagnosing cases of the first database. Additionally, its *diagnosis descriptors* created to interpret the knowledge of the neural network have been found to be similar to the knowledge elicited from experts for the same diagnostic problems. A drop in performance could be observed when the system was trained to diagnose cases of the second database, whose cases were collected without the supervision of an expert. Three other well-known databases have been used to evaluate the NN-CBR approach further. The hybrid system manages to solve problems that cannot be solved by the neural network on its own. Additionally, its indexing mechanism based on remindings provided by the neural network introduces a significant reduction in the number of case comparisons, if contrasted with a standard nearest-neighbour procedure.

By using these alternative knowledge representation and reasoning schemes, the hybrid NN-CBR approach suggests some solutions for common CBR problems (e.g. indexing and retrieval), as well as for neural network problems (e.g. the interpretation of the knowledge stored in a neural network and explanation of reasoning).

Acknowledgment

Thanks to Prof. John A. Campbell for his invaluable support and advice. His outstanding ability to always come up with a story related to the topics we had to discuss has been essential in guiding me throuout the three years of research I carried out at UCL. It has been a priviledge to work with such a remarkable man.

Thanks to Dr. Beatriz Leao for her help in providing the cases which were used in the validation phase of the work reported in this thesis, and for the comments on the results achieved.

Thanks to Dr. Eneida Mendonca for her additional collaboration in the collection of cases of Congenital Heart Disease.

Thanks to Jonathan Poole for his initial support in the beginning of my work at UCL.

Thanks to Mara Abel for her attention and concern, for the helpful comments and bibliographic references.

Thanks to Rafael Bordini for the friendship, the technical support and hospitality when hosting me indefinite times in London.

Thanks to my dear Gisella for being such a good friend, in occasions where I most needed one.

Thanks to my mother, my aunts, my brothers, and all the rest of my family, for always supporting me regardless of distance which separated us.

Thanks to Jean-Francois for motivating me, for giving me support in difficult moments, for his inexhaustible care and patience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents a new approach for integrating Case-Based Reasoning (CBR) with a Neural Network (NN) in diagnostic systems. In addition to being able to solve problems that cannot be solved by the neural network on its own, this hybrid approach proposes some useful solutions for common CBR problems, such as indexing and retrieval, as well as for neural network problems, such as the interpretation of the knowledge stored in a neural network and the explanation of reasoning. A hybrid NN-CBR model is introduced here, as well as its use in the development of a system for the diagnosis of Congenital Heart Diseases (CHD).

Case-Based Reasoning (CBR) is the field of Artificial Intelligence (AI) that investigates how *previous experiences* can be remembered and used in solving new problems, in an attempt to reproduce a reasoning process commonly observed in people.

Earlier work in Expert Systems (ES) (Hayes-Roth 1983) was different in that it focused on reproducing expertise by eliciting *knowledge* from a person, representing and using it in problem-solving. The usual attempt was to try to embody the knowledge of an expert in a set of rules, and apply these rules in a systematic form to solve new problems. Besides having some cognitive support, this idea was also commercially appealing, as machines could increase their expertise simply by augmenting their number of rules.

However, knowledge elicitation has proved to be a very difficult and time-consuming task (Hayes-Roth 1983). Furthermore, it has been noticeable that most experts do not rely only on a plain set of rules to solve new problems. Experts are often found to use cases to evaluate situations, suggest solutions and validate them (Klein & Calderwood 1988).

CBR has been introduced in the early 1980s as an alternative view of problem-solving and a more plausible model of human reasoning. Instead of relying solely on general knowledge of a problem domain, CBR was based on the use of specific knowledge of previously experienced, concrete problem situations (cases). The first memory model based on these principles proposed that a memory would be organised as memory organization packets (MOPs) whose functions were to store general knowledge as well as specific experiences of that general knowledge (Riesbeck & Schank 1989). This model was based on the Theory of Dynamic Memory (Schank 1982) which says that experiencing, understanding, remembering and learning cannot be separated from each other. When we are faced with a new experience, we try to understand it by remembering old experiences and comparing them with the new one. During this process the new experience is incorporated in our memory and associated with other experiences which we have had previously (learning).

IPP (Lebowitz 1983) and CYRUS (Kolodner 1983a; Kolodner 1983b) were the first systems to use MOPs as a cognitive model of a memory. IPP could read stories about terrorism, 'understand' them and build a memory of those stories. CYRUS had a similar use: to store and retrieve events in the life of Cyrus Vance, who was the Secretary of State of the USA. The system could understand stories collected from newspapers and answer questions about Vance's diplomatic meetings based on those stories. Following it, many other systems were developed using the same memory organization principles, such as CHEF (Hammond 1986), CASEY (Koton 1988), MEDIATOR (Kolodner & Simpson 1989) and PERSUADER (Sycara 1988). However, independently of the memory scheme used, the operation and the main principles behind CBR have remained the same: four '*REs*' steps are generally followed by the case-based systems, namely *REtrieve, REuse, REvise* and *REtain* (Aamodt & Plaza 1994; Watson & Marir 1994a). When a new problem is posed, the first step in the CBR cycle is the *retrieval* of similar cases. By comparing the new case with previous experiences, the case-based system must find which past experiences most resemble the new case. The solution suggested by the *retrieved* case(s) can then be *reused* for the new case. The assumption is that if a solution was applied successfully previously, there is a high probability that it can be used again in a similar situation. However, the old solution would usually have to be adapted to fit the requirements and constraints of the new problem. In the *revise* step, the solution composed for the new situation is tested for success. This is done by applying the solution to the real world, or by having it evaluated and repaired by a teacher when failures are detected. The last step in the CBR cycle is the storage of the new case, where its description and solution are indexed and *retained* for future use.

Many advantages have been introduced by case-based systems (Kolodner 1993). Firstly, CBR may be able to ease the task of knowledge acquisition (ie. eliciting knowledge from an expert and translating it into one of the knowledge representation formalisms) as most of the knowledge needed for solving new problems is contained in cases that have already been solved. Secondly, case-based systems can learn incrementally simply by appending new cases to their library of previous cases and indexing these cases for future use. CBR also seems to be a natural problem-solving method, which enables case-based systems to explain their reasoning relatively easily. Furthermore, CBR makes possible the composition of new solutions based on previous solutions used in old cases, which is generally much less complicated than building new solutions from scratch according to the requirements and constraints of the new case.

However, despite the relative success with which CBR techniques have been employed, they have also brought with them some disadvantages. For instance, in CBR systems reasoning is based on a set of cases that does not necessarily cover all the problem's solution space. Because of this, it is not guaranteed that an optimal solution for a given problem can be found. Although this last statement is true for every heuristic method, this is still a problem CBR researchers have to deal with. Other difficulties faced by case-based systems include the following:

- a case-based system might allow previous cases to bias it too much in solving a new problem;
- the most relevant experiences are often not considered when a new problem is being solved;
- non-case information (eg. rules), which may also be available, is ignored in a pure CBR system;
- for large systems, it might be a difficult task to manage an extensive case library, which does not happen with other generalization-based approaches.

Additionally, the idea that people use cases in problem-solving goes somehow against the observation that experts in typical application areas treated by AI methods reason by using generalised rules. AI practitioners have remarked that it is intuitive for experts to express their knowledge in the form of rules, and that this is not a new idea. In fact, some papyri have been found dating from the 17th century B.C where a series of diagnoses were described in an 'if ...

then ...' fashion (Crevier 1993). The counter-argument to this conviction is that representing knowledge with rules is different from actually using rules to reason. When using rules in problem-solving, it is often the case that rules have exceptions, and that some rules are also contradictory. When this happens, people usually start referring to previous experiences to try to explain these exceptions and contradictions.

The debate on which reasoning method is the most plausible, appropriate and/or efficient is a very long one. What is certain is that CBR and more traditional AI approaches to reasoning have their own advantages and drawbacks. In order to incorporate in one single architecture the benefits of both approaches and to minimise their drawbacks, CBR has been combined with complementary forms of reasoning, such as rule-based, model-based or neural networks. Our goal here has been to investigate what type of gain could be obtained from integrating CBR with neural networks to solve classification problems. The main reason for this choice was that by combining CBR with neural networks, we could benefit both from the logical and cognitive nature of symbolic systems and from the numeric, associative and self-adapting nature of the neural networks. Additionally, we wanted to continue our previous work in which neural networks were combined with symbolic systems (Reategui & Leao 1993; Leao & Reategui 1993; Leao, Reategui & Guazzelli 1995), as well as with CBR systems (Reategui & Campbell 1995; Reategui, Campbell & Borghetti 1995).

We present here a new approach for integrating Case-Based Reasoning (CBR) with a Neural Network (NN) in diagnostic systems. When solving a new problem, the neural network is used to make hypotheses and to guide the CBR module in the search for a similar previous case that supports one of the hypotheses. The knowledge acquired by the network is interpreted and mapped into symbolic *diagnosis descriptors*, which are kept and used by the system to determine whether a final answer is credible, and to build explanations for the reasoning carried out. The NN-CBR model has been used in the development of a system for the diagnosis of Congenital Heart Diseases (CHD). The system has been evaluated using two cardiological databases with a total of 214 CHD cases. Three other well-known databases have been used to evaluate the NN-CBR approach further. The hybrid system manages to solve problems that cannot be solved by the neural network with a good level of accuracy. Additionally, the hybrid system suggests some solutions for common CBR problems, such as indexing and retrieval, as well as for neural network problems, such as the interpretation of the knowledge stored in a neural network and the explanation of reasoning.

We present an overview of the research in hybrid architectures in chapters 2 and 3. The main reason for dividing the topic into two chapters has been that two types of hybrid architectures were related to our work:

- the ones combining CBR with other reasoning techniques;
- the ones combining neural networks with symbolic approaches.

While chapter 2 is devoted to the combination of CBR with other reasoning techniques, chapter 3 is devoted to the integration of neural network with symbolic approaches. Chapter 4 analyses with a greater level of detail some particular research which served as a basis for the development of the work reported in the present thesis. Chapter 5 introduces the new case-based architecture where CBR was combined with the CNM. A system addressed to Congenital Heart Disease (CHD) diagnosis is also presented in this chapter. This sytem was built with the collaboration of the Institute of Cardiology in Porto Alegre, RS, Brazil, which provided the cases to develop the case library of the system and test it. Chapter 6 presents validation results. Chapter 7 compares our NN-CBR approach with other related work. Chapter 8 presents conclusions and directions for further research.

Chapter 2

Combining CBR with other Reasoning Approaches

This chapter advances some criteria for analysing hybrid systems that integrate CBR with neural networks, rule-based or model-based approaches. Several systems are described and their features highlighted according to the criteria established.

2.1 Introduction

This premise that people rely on concrete previous experiences when solving new problem has been confirmed either by simple observation or (on some other occasions) by psychological experimentation. For instance, it has been observed that engineers rely on cases when trying to identify problems that occur in telephone switching networks (Kopeikina, Brandau & Lemmon 1988), that novices use previous examples to learn a new cognitive task (Ross 1989), or simply that people feel comfortable in using cases to make decisions (Klein & Calderwood 1988). However, determining how people remember pertinent cases at the right times has not been a trivial task. Moreover, how could computer programs simulate this behaviour? This has been one of the central problems in CBR, and has come to be known as the indexing problem.

2.2 Indexing in CBR Systems

Indexing a new case consists of creating appropriate pointers which designate under what circumstances the case may have a lesson to teach and, therefore, when it is likely to be useful.

Both manual and automated methods have been used in CBR index selection. Choosing indexes manually involves deciding under what circumstances a case will be useful, and identifying striking features which should make the case memorable. Although manual indexing is a difficult and time-consuming task, it may represent a good choice in some circumstances, especially when cases are complex and the knowledge needed for accurate case indexing is not concretely available, or it is too elaborate to be inserted in an indexing algorithm.

Choosing indexes with automated methods consists of building algorithms that can identify the circumstances in which a case can be useful. The main advantage of automated indexing over manual index selection is that it eases the knowledge acquisition process in which experts have to identify the features they use to recall past experiences, which is usually a non-trivial task. The methods for automated index selection most commonly used in pure CBR systems are:

- **Checklist-based indexing:** In this method the domain is analysed and the dimensions that tend to be important are computed. These are put in a checklist and all cases are indexed by their values along these dimensions. The system CHEF (Hammond 1986), a case-based planner whose specialty is to create new recipes, indexes known recipes through the following dimensions: *texture* and *taste of food, preparation method,* and *ingredients.* For example, a *Beef and brocoli* recipe could be indexed by CHEF the following way:

  Beef and brocoli:

  preparation: stir-fry
  ingredient: brocoli
  ingredient: beef
  texture: crunchy vegetable
  taste: savoury

  Although this method of automated index selection can be easy to implement, the method is only as good as the checklist used by the system builder. An incomplete checklist will result in insufficient indexing, while a checklist that does not discriminate between important and unimportant dimensions will result in overindexing and retrieval of too many cases.

- **Difference-based indexing:** How can a system determine that some indexed feature is not predictive? Or how can it appreciate that it is predictive? The answer relies on the identification of the similarities and differences among cases indexed in an equivalent fashion. The system CYRUS (Kolodner 1983a), whose main goal was to build a memory of events that happened in the life of Cyrus Vance, the US Secretary of the State at the time, found out that it could make no special predictions about meetings in which the main participant was a foreign minister because, by keeping track of similarities among several meetings with foreign ministers, it realised that they had nothing in common except the things common to all meetings. D*ifference-based indexing* tries to solve this problem by finding out features that can differentiate cases from one another so that at retrieval time best-matching cases can be selected from the library. But it is certain that not all features that are different across similar cases make useful indexes, and this is a problem *difference-based indexing* methods have to confront. For instance, it would not make much sense for CYRUS to index the meetings of the American Secretary of State by *room size,* as this would probably not allow any inferences to be made.

- **Explanation-based indexing:** *Difference-based* and *checklist-based indexing* provide a simple way for determining potentially predictive features to be used as indexes. But they have a major problem: they choose indexes based on a model of the kinds of features that are usually predictive but do not analyse cases individually for their predictive features. This causes two problems: extra features are often chosen for indexing that are not predictive in a particular case, and some features that are predictive in that case, but not in general, are not indexed. Because these methods do not always discriminate cases well enough from each other, retrieval based on indexes chosen by these methods results in many superfluous cases being retrieved. A separate ranking process must sort the cases that are recalled to determine which are the most appropriate. *Explanation-based indexing* methods are aimed at choosing indexes appropriately for individual cases. The reasoner attempts to explain why a given solution worked or failed to work:

1) Successful solutions (credit assignment): To create this type of explanation, a reasoner must keep track of the features found to be useful in achieving a goal. An example of this type of *explanation-based indexing* occurs in the system JULIA (Hinrichs 1988; Hinrichs 1989), which works in the domain of meal planning. When elaborating a meal for several guests, JULIA decides to make a tomato tart as the main dish. It chose this type of tart because it was summer and tomatoes were in season, and because a vegetarian friend was coming for dinner. 'Summer', 'tomatoes in season' and 'vegetarian guest' were the reasons for JULIA to select the tomato tart, therefore these are the elements used to index the case.

2) Unsuccessful solutions (assignment of blame): After the reasoner discovers it has made a mistake, it attempts to explain it. Then, the reasoner extracts from the explanation the concrete recognisable features of the situation that were responsible for the problem. In the next step, the reasoner generalises those features to the point where they still make sense in the explanation derived. An example of this type of *explanation-based indexing* can be given again from the behaviour of the system CHEF. When making the recipe of a strawberry soufflé, the results are not satisfying as the strawberries produced more liquid than the leavening could handle. The cause of the failure is attributed to 'too much liquid in *strawberries*', which can be generalised to 'too much liquid in *fruit*'. The unsuccessful case is then indexed by 'type of food = soufflé' and 'dish includes *fruit*'.

Despite the fact that *explanation-based indexing* relies on the analysis of the success or the failure of each particular case, there is still no guarantee that the indexes provided by the *explanation-based* methods will discriminate a case from similar ones. A combination of different indexing approaches can be used to provide a solution to this problem. For instance, an *explanation-based* method can be used to generate a set of useful indexes, while *difference-based* methods can be used to discard those features that do not discriminate.

Despite the possibility of combining an *explanation-based* method with other *checklist* or *difference-based* methods, explanation-based indexing has additional problems. Firstly, it can only be as good as the domain model from which explanations are drawn. Moreover, if the existing domain model is not complete enough, in certain situations the system would not be able to build any explanation for the faults identified, which would inhibit it from indexing the cases at hand.

Another problem with *explanation-based indexing* is that it indexes cases specifically for situations in which it is known that they are used. However, it is possible that a case can be useful in unanticipated ways, and indexing according to explanations alone may keep it from being remembered in these situations.

The indexing problems reported, as well as the wish to improve upon CBR systems, have led researchers to propose alternative architectures where case-based techniques have been combined with other reasoning mechanisms. The next sections present a general framework for describing hybrid CBR architectures, and describe several systems combining CBR with some other reasoning and/or learning technique.

2.3 Combining CBR with other Approaches

Although indexing has been one of the central problems in CBR, it has not been the only reason to incite researchers to combine CBR with complementary forms of reasoning, such as rule-based, model-based or neural networks. Other facts that have also influenced the construction of such hybrid architectures are:

- in order to represent human expertise more accurately, the specific knowledge stored in cases should be complemented by some form of generalised knowledge (Strube et al. 1995). For instance, students in US business schools learn from specific cases, as well as from general principles applicable to these cases. Therefore, cases and general knowledge should not be kept apart. An example of the practical use of such a combination of general and specific knowledge can be given by the congenital heart disease diagnostician *ChartD2* (Reategui, Campbell & Leao 1996a). The system, when solving a new case, uses its general knowledge to draw hypotheses and to guide the search for the most similar cases it has already 'seen'. The retrieved cases, representing specific knowledge, are then used to support one of the hypotheses and to justify the conclusion reached.

- managing large case libraries and getting the most out of cases may be a difficult task in CBR systems. Organising cases according to some generalisation principle has the possibility of giving the user a clearer view of what types of cases exist in the library, and reducing the number of case comparisons when a best-match is looked for. The case-based tool REMIND, for instance, organises the case library by employing user-defined attributes to create an initial prototype hierarchy, which is then partitioned further by an inductive method. The final hierarchical structure is efficient for case-browsing and reduces the number of case comparisons during case-matching.
- previous cases may influence a CBR system in different directions without giving it many hints on which cases to consider as more important. This problem can be partially solved by good indexing schemes which enforce the retrieval of relevant cases only. However, in large case libraries, even efficient indexing mechanisms may have problems in indicating exclusively pertinent cases. In such circumstances, it may be advisable to have only a reduced set with selected cases for consideration in the case-matching process. For instance, Surma & Vanhoof (1995) use an algorithm to partition the cases of the library into standard and exception cases. The standard cases are used to generate rules, while only the exception cases are used in the case-matching process.

Several other techniques have been used to combine CBR with other reasoning approaches. In the project INRECA (Manago et al. 1993; Auriol et al. 1994) a general framework for combining induction with CBR in classification tasks has been proposed and implemented in the generic tool that combines the systems PATDEX (CBR) and KATE (induction). This integration has been based on the understanding of the advantages and weaknesses of each approach. Four levels of integration between induction and CBR were identified in the project, each level providing certain advantages that could be suitable for one type of application or another. The integration levels defined in INRECA were modified and further specified here in order to describe other mixed paradigms combining CBR with different reasoning mechanisms[1]:

[1]The levels defined originally in INRECA are: toolbox (the two mechanisms work in an independent fashion); co-operative level (one mechanism uses the results of the other to improve its performance);

(1a) *independent approach*: the two reasoning mechanisms work side by side in an independent fashion. They are activated to solve the same problem, but they may reach equivalent or different answers. When reaching different answers, an external mediator, or the user himself, decides which answer to take. For instance, a rule-based and a CBR system can be used independently to identify malfunctions in an engine. When different answers are provided by each system, the user himself or an external mechanism decides which answer to take. This may be useful for the comparison of two different techniques in solving a particular problem, with the advantage that two existing systems can be employed.

(1b) *back-up approach*: one of the mechanisms starts the reasoning and transfers it to the other mechanism when it cannot complete the process. The main reason justifying this approach is that alternative results can be presented to the user when the reasoning mechanism commonly used to solve the problem cannot reach a plausible answer. For example, the system CASEY (Koton 1988), a cardiac disease diagnostician, starts its reasoning process by first activating a CBR module. If no answer is provided by this module, the system attempts to solve the same problem by activating the Heart Failure Program, an earlier model-based expert system.

(1c) *interactive approach:* one of the mechanisms starts the reasoning process and transfers some intermediate results to the other mechanism, which proceeds with the reasoning from that point to the end. This approach is frequently used in the design of alternative indexing methods. For instance, in a system for fault diagnosis, model-based reasoning may be used to estimate the probable location of a faulty component. The hypotheses generated can be then used by a CBR module to retrieve similar previous cases (Feret & Glasgow 1993). Another alternative to this approach is that a higher level of interaction can be established between the two reasoning mechanisms, where the responsibility for the reasoning is transferred

---

See Riesbeck & Schank (1989) for a more detailed explanation of MOPs and a description of several systems implemented using these memory structures.

workbench level (individual models and capabilities of each mechanism are combined); seamless integration (the most relevant parts of the two mechanisms are mixed in a single system).

several times until the system reaches a final conclusion.

(1d) *unified approach:* the two reasoning mechanisms are completely integrated by having certain features of each of them combined in a single system. This approach is most commonly used in the combination of CBR with neural networks, with the main goal of creating efficient indexing and retrieval mechanisms. For instance, the system INSIDE (which was developed for the use of Singapore Airlines), keeps cases as intermediate nodes of a neural network. Case indexing and retrieval in this hybrid architecture are a consequence of the learning and activation functions of the neural network, respectively.

The four levels of integration described above are illustrated in figure 2.1.

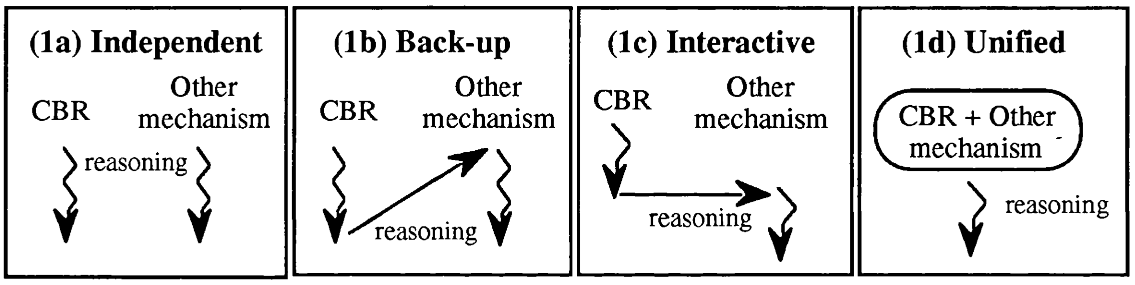


Figure 2.1: The four integration levels for hybrid CBR systems

For each of these levels, other criteria can also be employed to detail further the type of integration used: The levels of integration (1a), (1b) and (1c) can employ one of two types of general knowledge:

(2a) *compiled knowledge:* general knowledge is the result of the pre-processing of existing cases (e.g. decision-trees or rules inductively generated from cases, neural networks);

(2b) *elicited knowledge*: general knowledge can represent the knowledge of an expert, knowledge found in books or simply commonsense knowledge (e.g. rules or graphs learned through a knowledge acquisition process).

For level (1d), the most natural choice is to use *compiled* general knowledge, by having the same cases of the library pre-processed to originate general knowledge. However, it is also possible that a system would, for instance, use cases as the consequents of expert rules (*elicited knowledge*), and have them selected as near matches when the rules are fired.

Different levels of priority can be assigned to the reasoning mechanisms. This feature applies mostly for architectures where the two mechanisms are separate, working in a somewhat competitive way, as in (1a) and (1b). For architectures where the co-operation between the components is higher, it becomes difficult to establish the level of priority between them.

(3a) giving *priority to CBR*: the other reasoning mechanism serves as an alternative device when no final answer can be provided by the CBR system;

(3b) giving *priority to the other reasoning mechanism*: the CBR system is used to produce a final answer when the other reasoning mechanism fails to solve the problem addressed.

Giving *priority to CBR* can be a good alternative for applications in domains with a weak model (or a limited body of rules), where cases are usually of primary importance in the reasoning process. In contrast, giving *priority to the other reasoning mechanism* can be more suitable for applications where there is a strong domain model with well-established rules, or where cases are sparse or difficult to collect.

For level of integration (1c) and (1d), the co-operation between CBR and the other reasoning mechanism can have different purposes:

(4a) *general knowledge supporting CBR:* general knowledge can be used to assist in a CBR task. For example, rules extracted from cases can be used to index and retrieve cases in the library (Zeleznikow, Hunter & Vossos 1993);

(4b) *cases complementing general knowledge:* cases can support the reasoning performed by the mechanism based on general knowledge. For example, cases can be used to deal with missing information in a consultation of a decision-tree (Auriol et al. 1994);

(4c) *mutual support:* cases and general knowledge can support each other in the reasoning process. For instance, a rule may support the selection of a particular case as the most similar, and an existing case can support the firing of a rule (Rissland & Skalak 1989).

Because of the smaller amount of interaction existing in architectures of the type (1a) and (1b), it is difficult to determine the level of co-operation between the two mechanisms.

For levels of integration (1a), (1b) and (1c), the knowledge embedded in cases and in the generalisations can be:

(5a) *equivalent* and address the same problems;

(5b) *complementary* and address different problems.

To illustrate item (5a), a system that diagnoses car malfunctions could store previous cases of car problems, as well as rules specifying the problems related to certain collections of symptoms (this is the approach generally used to combine CBR with Model-Based Reasoning (MBR), which refers to the use of general causal knowledge, usually to describe well-understood causal devices). To exemplify item (5b), a similar system diagnosing car malfunctions could have a set of rules that, instead of identifying the car problems themselves, would be used to determine other associated symptoms that could help the system to retrieve the most similar previous case from the library.

For *unified* architectures, it is usual that both cases and general knowledge would be *equivalent*, as they address the same problem. The reason for this is linked to the fact that the usual choice of *unified* approach employs *compiled knowledge*, ie. the cases pre-processed to originate general knowledge are the same as those used in the reasoning process.

These characteristics provide a framework for the analysis of hybrid CBR systems that combine CBR with other reasoning approaches. Table 2.1 lists several such systems and their main features, and the next section comments briefly on each of them.

Systems / features	Level of integration	Type of general knowledge	Priority in the reasoning process	Form of support	Knowledge embedded in cases and in generalisations
CABARET	interactive	elicited	*	mutual	equivalent
GREBE	interactive	elicited	*	mutual	complementary
IKBALS	interactive	compiled	*	general knowledge supports case retrieval	equivalent
INRECA	various	compiled	various	various	equivalent
REMIND	unified	*	*	general knowledge used to structure the case library and to support case retrieval	*
BUBE/CcC+	interactive	compiled	*	general knowledge supports case retrieval	equivalent
Surma & Vanhoof (1995)	back-up	compiled	rules	*	equivalent
CASEY	back-up and interactive	elicited	CBR	general knowledge supports case retrieval	equivalent
CABATA	interactive	elicited	*	general knowledge supports case retrieval	complementary
ARAMIIHS	back-up	elicited	rules, then CBR, then MBR	*	equivalent
Feret & Glasgow (1993)	interactive	elicited	*	general knowledge supports case retrieval	equivalent

Table 2.1: Comparison of CBR systems

* classification does not apply

2.3.1 Combining CBR with other Symbolic Reasoning Mechanisms

An approach that has become rather popular within the CBR community is the use of rules in cooperation with CBR. Many of the systems of this type are legal applications, where the partitioning of the knowledge into rules and cases was a natural alternative, as in CABARET[2] (Rissland & Skalak 1989; Skalak 1989), GREBE (Branting 1989) and IKBALS (Zeleznikow, Hunter & Vossos 1993). These three systems combine CBR and rules in an *interactive* approach, as in fact most hybrid CBR architectures do.

CABARET (Skalak 1989) has been conceived as a domain-independent shell, and applied in statutory interpretation, ie. in the determination of the meaning of a legal rule by analysing its terms and then applying it to a particular set of facts. CABARET integrated a CBR and a rule-based module using a central control scheme, as defined in Skalak (1989)[3]. The system worked with 30 types of expert heuristics to control and interleave reasoning with rules and reasoning with cases. The system's architecture was characterised by the following:

- there are two co-reasoners (case-based and rule-based);
- each co-reasoner has an independent mechanism to report the results achieved, or some intermediate state of the reasoning;
- there is a controlling process that uses the set of heuristics and the reports provided by each reasoner to decide how the system as a whole is to proceed.

[2]Although CABARET has been conceived as a domain-independent shell, its first application was in the domain of law.

[3]Four different ways of integrating CBR with Rule-Based Reasoning (RBR) have been identified in Skalak (1989), according to the type of control used to co-ordinate the two reasoning mechanisms: Co-equal reasoners with invocation by a separate control module; Co-equal reasoners with two-way invocation (the control between the two reasoning mechanisms is dispersed between them); CBR-dominant (the RBR mechanism may be called by the CBR module, but not the opposite); RBR-dominant (the CBR mechanism may be called by the RBR module, but not the opposite). This classification scheme has also influenced the classification presented here. However, we were more concerned with the type of interaction between the two reasoning mechanisms, whereas Skalak (1989a) has focused mostly on the type of control used to manage them.

Legal rules and cases addressed the same types of problem. However, although being combined in an *interactive* approach, rules and cases might contradict each other. For instance, a rule may say that an attitude should lead to punishment, and the case library might contain cases contradicting it (or supporting it). This feature enables CABARET to reason from different points of view (e.g. the prosecuted or the prosecutor). Thus, even with the same set of rules and cases the system can reach different conclusions. The main contributions of the accompanying research have been the establishment of a framework for the development of systems integrating CBR and rules, as well as the definition of the set of heuristics for a central control module to co-ordinate the reasoning process.

The system GREBE (Branting 1989) has also been designed for the domain of law. It too integrates CBR and RBR in an *interactive* approach, as in CABARET. GREBE uses both legal and commonsense rules, in addition to category exemplars (previous cases) to determine the classification of new cases. While exemplar-based explanations are used to bridge the gap between new case descriptions and generalisations, domain theory in the form of general rules is used to explain the equivalence of new cases and category exemplars. Generalisation-based reasoning and CBR are considered complementary approaches in GREBE, where each of which is necessary for the success of the other. It has been demonstrated that some explanations that GREBE produces could not be produced by other methods limited exclusively to generalisation or case-based reasoning.

The system IKBALS (Zeleznikow, Hunter & Vossos 1993) has as a main goal to investigate ways of reducing the problems associated with modelling law using only a rule-based approach. Among the main problems identified in the use of rules in legal expert systems are the difficulties in representing open-textured legal predicates using rules, as well as the deficiencies found in the explanations produced by traditional expert systems developed in this domain. Differently from CABARET and GREBE, IKBALS uses an inductive algorithm (Induce2) to produce a decision-tree from the available cases. This tree is later converted into production rules that are then used by the system to locate relevant similar cases. Another difference between IKBALS and CABARET is that the latter uses a central control to perform the integration between the rule-based and the CBR modules, while IKBALS uses a distributed control (as defined in Skalak (1989)).

Another important approach to combining inductive learning methods with CBR has been delineated in the INRECA project (Manago et al. 1993, Auriol et al. 1994), which we have already mentioned in the introduction of this chapter. In the project, advantages and weaknesses of CBR and induction were analysed and distinct levels of integration were proposed in order to minimise drawbacks and use the best features of each approach. In each level of integration, different alternatives are possible for combining CBR and induction. For example, cases can be used to minimise the problem of handling missing values during consultation in a decision-tree (*CBR complementing general knowledge*); and decision-trees can be used to perform consistency checks in the case library (*general knowledge supporting CBR*). Many applications in different domains have been tested in the INRECA project, including the maintenance of aircraft engines, the estimation of risk factors for car insurance companies and the selection of hotels according to travellers' preferences. A recent extension to the project has been the incorporation of the INRECA tree (Auriol et al. 1995) into the system. This tree is based on both decision-trees (Quinlan 1986) and k-d trees (Friedman, Bentley & Finkel 1977). The main advantage of the new INRECA tree is that it can be used either as a k-d tree in the case-based reasoning process (e.g. in case indexing and retrieval) or as a decision-tree in the induction process (e.g. for case generalisation).

Similarly to the PATDEX/KATE tool developed in the INRECA project, the system REMIND (Barletta 1994) is a generic tool for developing systems that combine CBR and inductive processes. However, REMIND is based on a *unified* approach where the cases are part of an indexing/retrieval scheme divided into three levels: a prototype hierarchy, an inductive indexing structure and the weighted nearest-neighbour algorithm. At the top of the indexing structure is the prototype hierarchy. This hierarchy is built according to attributes selected by the system developer that serve as an initial support for partitioning the available cases (for indexing and retrieval purposes). The inductive method is then used to partition the prototypes of the lower branches of the hierarchy further. When a new case is presented to the system, it compares the new case with the prototype structure and then retrieves cases under the most specific prototype that matches the new case. Once the best prototype is matched, the system follows the inductive indexing structure created for that prototype. The cases collected in this last step are then passed to the nearest-neighbour matching algorithm for a final ranking procedure. REMIND has already been used in the development of systems in different application areas, such as in legal decision-making, in medical outcome analysis, and in help desks.

In Bamberger & Goos (1993) another different integration method is presented where a case-based and an inductive learning system (CcC+ and BUBE, respectively) are combined in an *interactive* architecture. BUBE is used to learn rules that are later employed to reduce the number of cases that are analysed by CcC+ when a problem is being solved (general knowledge supporting case retrieval). The learned rules generate a set of possible solutions, and provide the cases corresponding to these solutions to the CcC+ system for a more detailed similarity computation. One of the main advantages of BUBE/CcC+ is that the response time of the inference process is minimised by the integration of CBR and rules, while the performance and the quality of the explanations provided are kept at a good level.

Surma & Vanhoof (1995) also use induction to learn rules from cases. However, their approach to reasoning is different in that their architecture has a *back-up* level of integration which gives *priority to the generalised knowledge* expressed in the rules. This reasoning scheme is based on the following assumption (from Riesbeck & Schank (1989)): ''When an activity has been repeated often enough, it becomes rule-like in nature. We do not reason from prior cases when well-established rules are available''. The algorithm below has been conceived according to this assumption:

```
If a new case is covered by some rule,
    then apply a solution from a rule with the highest priority,
    else adapt the solution from the most similar case.
```

In this architecture the case library is split into two disjoint sets, one containing exception cases and the other containing standard cases. The set of standard cases is used by an induction algorithm to produce a rule-base. When a new case is presented to the system, this rule-base is used to classify the case. If no existing rule applies, the case is classified by the nearest-neighbour algorithm using the exception cases. This problem-solving approach has been tested with relative success on a few different domains. However, the method of giving *priority to the generalised knowledge* contained in the rules might not be suitable in other domains where cases play an important role, as in the domain of law.

Another hybrid CBR approach is the combination of CBR with Memory-Based Reasoning (MBR). One of the best-known systems that uses such an approach is CASEY (Koton 1988). CASEY integrates CBR with the Heart Failure Program, a model-based expert system that deals with patients with cardiac diseases. CASEY's memory organisation is based on CYRUS (Kolodner 1983a). It contains descriptions of previous cases as well as generalisations derived from similarities between cases. The descriptions of cases are made out of input features such as symptoms, test results and medical history, as well as solution data, such as the causal explanation for the patient's symptoms, therapy recommendations and outcome information. The steps below summarise the method used by CASEY to solve a given case:

- retrieval of a similar case from the library;
- evaluation of the differences between the new and the previous case using the knowledge existing in the Heart Failure Program, with the main purpose of ruling out answers that present inconsistencies;
- If no match is found for the new case, the Heart Failure program can be used to solve the problem.

This architecture uses mainly a *back-up* level of integration with *priority given to the case-based component*. However, the fact that the Heart Failure Program provides similarity metrics for the CBR mechanism reveals a higher level of interaction between the two reasoning mechanisms than that found in more standard *back-up* approaches. The main advantages introduced by CASEY were the use of model-based knowledge to control the quality of a match performed by the CBR system, and the use of the Heart-Failure Program to solve diagnostic problems that could not be solved by the CBR mechanism.

The system CABATA (Lenz 1993) also combines CBR and MBR in an *interactive* architecture conceived for planning tasks. The system was tested in a travel planning application, where a previous holiday trip had to be retrieved from memory according to some preliminary specifications established by the user. In addition to these specifications, the system uses two types of rules to help it find a similar previous holiday: rules that can increase or decrease the importance of certain features (e.g. if holiday type is city then season is less important); and rules that contain domain-specific knowledge to build constraints (e.g. if region is sea then region must not belong to mountains). One peculiarity of CABATA is that the reasoning is conducted in a way where the user interacts with the system, having the chance to interfere in a solution by modifying conditions and constraints.

Feret & Glasgow (1993) present another architecture combining CBR with MBR. The architecture has been applied to the diagnosis of faults in two different devices, namely a robotic system called Fairing servicing subsystem, and a Reactor building-ventilation system. The approach to modelling the devices has been based on the Automated Data Management System, which describes a device in terms of a hierarchy of components or groups of components. The hybrid architecture uses an *interactive* approach where an estimate of a location of a component whose failure explains the observed symptoms is first generated using the MBR techniques (*general knowledge supporting case retrieval*). These hypotheses are then used to retrieve previous cases. The main role of CBR in this system is to refine the possible answers given by the MBR module and to assist an operator in the final stage of a diagnostic session. The tests executed in the two experiments indicate that the performance of the hybrid approach is better than the MBR approach used on its own, and that CBR can minimise the impact of using partially incorrect models.

In ARAMIIHS (Macchion & Vo 1993), MBR, CBR and rules are combined to detect problems in a vehicle equipment bay that schedules the launching of a flight by computing guidance actions, telemetry, emissions and other types of evidence. The MBR mechanism of the system has three purposes: describe technical, causal and structural aspects of the vehicle equipment bay. The diagnostic capabilities of this mechanism are activated when CBR and the mechanism of rules fail to provide a satisfying solution to a given incident. The CBR mechanism is only activated when rules cannot solve the problem. Indexes connecting from one to several symptoms to a case have either a positive or negative influence on the associated diagnosis. This predominance in the reasoning process characterises ARAMIIHS as a mainly *back-up* architecture with a scheme of mixed priority levels: priority is given firstly to rules, then to CBR, then to MBR. The authors claim that from a cognitive point of view the system mimics the expert's reasoning process: if an incident occurs frequently, he can recognise the situation automatically (use of rules). If the problem has already occurred previously, he can recall its past global or partial solution (use of CBR). Otherwise he must activate a heavier process based on a technical representation of the equipment to be diagnosed (use of MBR).

2.3.2 Combining CBR with Neural Networks

Another approach that is becoming more common is the combination of CBR with neural networks. The majority of the architectures integrating these two reasoning mechanisms has a *unified* level of integration. This frequent choice of architecture is due to the fact that it can provide good solutions for CBR indexing and retrieval problems. For instance, cases can be represented in intermediate nodes of the neural network, being indexed by the network learning mechanism, and retrieved by the network consultation mechanism.

The architecture presented in Becker & Jazayeri (1989) has been developed to deal with design problems, where the previous case most similar to the case being solved has to be retrieved so that its design solution can be adopted directly for the new case, or modified to fit new requirements .

To perform this task, the system uses a neural network composed of three layers: an input layer where features are described, an intermediate layer connected to the input layer where old cases are represented, and an output layer where the design choices are kept.

Three types of knowledge are used to select the most similar cases: knowledge about the problem (input features); knowledge about old cases (old cases' features and the design solutions adopted); meta-knowledge about criteria for selecting new cases (importance of each attribute). The network also includes modification rules that are used to change the importance of certain attributes according to the values present in other attributes.

Regarding the level of integration between case-based and neural-network components, this architecture has a *unified* approach. The cases themselves are part of the neural network, and case-retrieval is carried out by this hybrid structure. This type of architecture is interesting from the point of view that a neural network can use old cases to support a decision on a design solution. However, because the neural network is used to connect all the previous cases to their solutions and descriptions (case features), the neural network is not used to form generalisations. Thus, when solving a new problem, it has to compare each case individually with the current situation and select the one that appears to be the most similar (thus, many of the inconveniences of the nearest-neighbour algorithm can also apply to this system, such as a processing time that increases linearly with the number of existing cases, and a possible lower accuracy rate than other algorithms (Barletta 1994)).

Thrift (1989) also combines neural networks and cases in a *unified* approach, where the neural network is used for case filtering, ie. selecting the most relevant cases in a case library given a set of input values. By using a set of training cases different from the case library, this system manages to form generalisations while keeping relevant previous cases for future reference.

The neural network used in this architecture has been conceived with three layers: an input layer composed of 'factors' (features and feature values); an intermediate layer composed of previous cases; and an output layer composed of actions (solutions adopted in the previous cases). Therefore the architecture proposed by Thrift is also characterised by being *unified*, as cases are part of the neural network. The backpropagation algorithm is used to adjust the weights connecting factors to previous cases. Seeing it from a different point of view, the algorithm is used to establish a similarity measure between cases. The case selection is also influenced by the presence of hidden factors that are computed in accordance with the input factors. The backpropagation algorithm determines changes in the function of the action units, according to given input factors. The author remarks, however, that many of the current learning algorithms could equally well be used to adjust the weights that appear as parameters in the system.

When presented with a new problem the neural network indicates the most similar case in the library by taking its maximally-responding case unit. The solution adopted for that case is applied directly to the new case or modified to conform to some new conditions.

This hybrid neural-network approach has the advantage of being able to refer to previous cases when giving a solution to a new problem. In addition, because a training set different from the case library is used to train the neural network, its generalisation capabilities are bigger than those of Becker & Jazayeri (1989).

In Myllymaki & Tirri (1993), another example of a *unified* architecture combining CBR with neural networks is presented, where cases correspond to certain nodes in the neural network. In the architecture proposed, a CBR system using Bayesian probabilistic reasoning is implemented in a connectionist network. The system is addressed to case matching and adaptation problems.

The neural network is composed of three layers, the connection weights between nodes of different layers being computed using probability theory. The first layer of the network contains one node for each possible attribute value a_{ij}. The output of these nodes is either entered by the user or calculated by the a priori probability of the presence of the value. The second layer has several groups of nodes, each group representing one attribute. The weight $W(A_i)$, connecting a value node a_{ij} to an attribute node A_i, is $P(A_i = a_{ij} \mid C = c_k)$, ie. the conditional probability that the attribute A_i has the value a_{ij}, given an observation from class c_k. The activation value of the nodes in the second layer is calculated by the weighted sum of the values coming from the first layer. In the third layer there is one node for each existing class. The activation of these nodes is calculated as a function of the weighted product of the values coming from the previous layer.

This is another example of a *unified* architecture combining CBR with neural networks, where cases correspond to certain nodes in the neural network. The major advantage of this architecture is that it provides a sound theoretical explanation for the case-matching process via probability propagation. This architecture is similar to those of Becker & Jazayeri (1989) and Thrift (1989), as they all represent the entire set of previous cases in the intermediate layer of the network. One of the main problems in this type of approach is that in the presence of noise it might be disadvantageous to use all the cases in the library. In order to improve the performance of the system, an alternative method for disposing of noisy cases may be used.

Grow and Learn (GAL) (Alpaydin 1991) uses a neural network with a topology similar to those of Becker (1989) and Thrift (1989). However, GAL keeps only representative cases in the intermediate layer of its three-layered network. It uses the prototype-based incremental principle, where a given class of objects is represented by the accumulation of relevant exemplars of the class and the modification of other class representations. GAL is composed of three layers, where the first layer is formed by input nodes, the second layer by exemplar (or prototype) nodes and the third layer by nodes representing classes. In the training phase, for each new case presented to the network, a winner-take-all procedure selects the closest exemplar in the network. When no exemplar exists for a class, a new node has to be created in the third layer of the network and connected to its new exemplar. Correct answers do not originate any changes in the network. For incorrect answers, new exemplar nodes have to be created and connected to their corresponding classes. In order to decrease the complexity of the network, a *sleep phase* has been introduced here. In this phase, one of the exemplar units is selected at random. The input vector is set equal to the weight vector of the chosen exemplar and the response of the network is computed. The response is the class to which that exemplar is connected. Then, that particular exemplar is disabled and the response is computed once more, this time without that unit. If the class found in the two cases are the same, that exemplar is removed. This procedure reduces the number of exemplars that are close to class boundaries, but not as close as other exemplars.

The learning method adopted by GAL does not consider the densities for the classes represented. Because of this, GAL can be compared with the nearest-neighbour algorithm where, instead of using all cases for comparison with new cases, only representative cases are kept.

The architecture of ARN2 (Azcarraga & Giacometti 1991) is similar to the architecture of GAL. Once again, the first layer (or the input layer) contains nodes representing attributes; the third layer (or the output layer) contains nodes representing classes. The second layer (or intermediate layer) contains nodes corresponding to prototypes, which are represented by reference vectors. A reference vector with n elements can be considered as a point in n-Euclidean space, possessing an influence region with a predefined radius. In the learning phase, when the system gives a wrong answer, the influence region of the activated prototype node is decreased in order to exclude the misclassified example. This is one of the main differences between ARN2 and GAL. Another important difference between the two models is that ARN2 does not have a sleep phase where ineffective prototype nodes are removed from the network.

The system INSIDE (Lim et al. 1991) uses a very similar approach to that of GAL and ARN2. It keeps only representative cases as part of its neural network, and its learning algorithm is based on the same principles as the algorithm used by ARN2. INSIDE was developed for the use of Singapore Airlines with the purpose of giving technicians support for diagnosing problems in the Inertial Navigation System used by their aircraft. Its neural network has three layers: an input layer that keeps symptoms and test results; a hidden layer fully-connected with the input layer and where the representative cases are kept; an output layer sparsely-connected to the hidden layer where each node corresponds to a type of fault that can be diagnosed by the system.

A clustering-based learning algorithm is adopted to train the neural network. This algorithm creates clusters for representative cases according to their features. Then, for each new case, a distance is calculated between the existing cluster centres and the new case. If the new case falls correctly into one of the clusters' zone, the case is only marked 'covered'. When it falls incorrectly into a cluster zone, the radius of attraction of that cluster is diminished. Cases that do not fall into any zone have a new cluster created for them.

A consultation in this system starts by the computation of the Euclidean distance between every cluster centre and the feature vector of the new case. The smallest distance shows the closest cluster, which corresponds to one of the nodes of the hidden layer, ie. a representative case. This representative case is connected to certain fault diagnoses that are assumed to be present in the new case too.

Furthermore, INSIDE contains a Flowchart mechanism that is used to provide solutions when the neural/CBR mechanism fails to give an answer. By using this other mechanism, the system attempts to cover both typical and more atypical cases. The main difference between INSIDE and ARN2 resides in the particularities of the methods used for learning prototypes and adjusting their influence regions.

The Prototype-Based Indexing System (PBIS) (Malek 1995) is based on ARN2, but its main purpose is to improve the performance of ARN2 by keeping both prototypical and non-prototypical cases. PIBS has a *unified* architecture with a memory divided into two different levels. The first level is the intermediate layer of the ARN2 network containing prototypical cases. The second level is a flat memory in which similar cases are grouped together in zones. Each zone of similar cases is connected to the closest prototype of the same class. Additionally, there is an atypical zone which stores boundary cases that fall into uncertain regions.

When a new case is presented to the network, if only one class is activated, the prototype with the highest output is selected. When more than one class is activated, the system selects the memory zones associated with the activated prototypes and retrieves the most similar case from these memory zones. When no prototypes are activated, the system looks for similar cases in the atypical memory zone.

The main difference between PBIS and other prototype-based incremental neural networks is that PBIS uses both prototypical and boundary cases in the reasoning process. It provides an interesting approach to dealing with typical and atypical situations. The prototypical knowledge of the intermediate layer is used to classify the more ordinary cases, while less common problems are solved with the help of the other memory level (typical and atypical zones). The consequent switching from neural networks to case-based search gives this model some of the features of *back-up* architectures as well.

When we compare prototype-based incremental neural networks with other models based on inductive methods, incremental learning in the networks seems to be simpler. In PBIS, for example, learning means adding cases to the memory zones and modifying the influence regions of the prototype nodes. In systems using inductive methods, learning usually implies the rearrangement of large parts of hierarchical structures.

All the systems combining CBR with neural networks presented above have an architecture with a *unified* level of integration where the network is used mainly for case retrieval purposes. Table 2.2 shows a summary of the main characteristics of these systems according to the framework detailed at the beginning of this chapter.

	Level of integration	Type of general knowledge	Priority in the reasoning process	Form of support	Knowledge embedded in cases and in generalisations
Systems combining CBR and neural networks	unified	compiled	*	integration for case retrieval purposes	*

Table 2.2: Main features of systems combining CBR with neural networks

* classification does not apply

2.4 Difficulties and Inconveniences with Hybrid CBR Approaches

Most of the problems in diagnostic CBR systems are related to indexing, ie. how to remember the appropriate cases at the right times. The CBR architectures that combine case-based methods with other symbolic approaches provide some solutions for such problems. For instance, decision-trees can be used as a means of case organisation/retrieval, while production rules can be used to locate relevant similar cases. One of the main inconveniences with the decision-tree mechanism is that for classification tasks that become more intricate, the knowledge represented in decision-trees is difficult to understand (Quinlan 1993; Winston 1992). Furthermore, updating decision-trees may require a great deal of computation (sometimes the rearrangement of the entire tree), which can make learning a hard task.

In contrast, the architectures that combine neural networks with CBR usually have more ‘elegant’ learning mechanisms. The incorporation and indexing of new cases in the library is done in a natural way. However, using the network to learn generalisations through previous cases produces a representation of knowledge that is often difficult to understand. And as stated by Sammut (1995), a representation that is opaque to the user may allow the program to learn, but a representation that is transparent to the user also allows the user to learn.

Here, we have focused on the development of a case-based model that would represent classificational (or diagnostic) knowledge in a format that is easy to understand. However, we also wanted the architecture to provide the users with good learning and explanation capabilities. The solution conceived was the following:

- the Combinatorial Neural Model has been used to provide the system with learning capabilities. The network not only identifies important features and combinations of features that can be observed frequently in certain diseases, but also provides indexes for the search for similar previous cases when a problem is being solved;
- diagnosis descriptors have been used to make explicit the opaque classificational knowledge of the neural network. These descriptors have a simple and readable structure, which enables the user to understand the diagnostic knowledge employed by

the system in the reasoning process.

- CBR has been used as a mechanism for retrieving similar previous experiences that can support a diagnostic hypothesis. Using previous experiences to support a result that has been reached can be justified by the fact that it is intuitive (Kolodner 1993, page 27) and that people are more ready to accept solutions that are grounded explicitly on previous cases (Georgin et al. 1994).

Chapter 5 gives further details on how each of these techniques has been used in the hybrid architecture, and chapter 7 explains how our work is differentiated from other previous research.

Chapter 3

Related Research: Integrating Symbolic and Connectionist Processing

This chapter complements the previous one by presenting some related research which investigate the benefits of combining neural networks with other symbolic approaches. The major reason for the interest in this other type of hybrid architecture is that in certain circumstances one single approach cannot provide all the necessary resources to represent knowledge and reason in a given domain. Although it may be desirable to have models in which the components are as simple and homogeneous as possible, from an engineering perspective having a connectionist model with a hybrid configuration may well be the only solution for a given application (Lange 1992). This chapter presents a brief introduction to symbolic and connectionist processing. The strengths and weaknesses of each approach are highlighted, and their complementarity and cognitive appeal are discussed. Following that discussion, several connectionist/symbolic systems are examined.

3.1 Introduction

Despite sharing the same philosophical foundation, i.e. that cognition or thought processes can, at some level, be modelled by computation (Honavar 1995), neural and symbolic approaches were based on very different architectural concepts. While symbolic systems were mainly representation-oriented, connectionist systems have been process-oriented. Thus, one of the main goals of symbolic AI has been the definition of the most appropriate structures to express the knowledge of a particular application domain, as well as the methods needed for handling these structures. In contrast, neural-network scientists were more concerned with creating connectionist architectures in which intelligent behaviour would emerge from the interaction of a large number of simple processing units.

One of the first connectionist models, introduced in the early 60s, was the Perceptron (Rosenblatt 1962). Its learning and adaptation capabilities raised a great deal of interest and expectations. In 1969, however, Minsky and Papert (1969) pointed out a number of limitations in the model, which had serious consequences in the development of the field in the following years. In 1974, Paul J. Werbos created a procedure that could adjust the weights of a neural network and that did not have the limitations of the Perceptron. This procedure, known as backpropagation, was rediscoverd in the 80s and popularised by Rumelhart, Hinton & McClelland (1986a). It consists of a learning technique in which the network is able to adjust its connection weights based on a given set of examples containing an input pattern and a target output pattern. This period could be considered as seeing the rebirth of neural networks, when two lines of research became distinct. The first was more concerned with the cognitive implications of connectionist models and would try to deal with higher-level problems such as representing knowledge or implementing production systems with neural networks (Touretzky & Hinton 1987). The other line focused more on lower-level pattern-recognition problems, such as in signal processing, speech and image recognition (Simpson 1990).

However, despite the efforts in trying to use connectionist systems in higher-level tasks, connectionist systems suffered strong criticism for their lack of ability in performing certain cognitive tasks (Fodor & Pylyshyn 1988). The argument has been that neural networks can learn representations that model the statistical properties of a set of cases, but that they cannot learn structural relationships that may exist between concepts in the domain knowledge. The reason for this is that connectionist models acknowledge only causal associations as a primitive relation among nodes in a network. Traditional work in symbolic AI, in contrast, acknowledges not only causal relations among the concepts represented, but also structural relations. Thus, in connectionist models the relation A->B can only mean that some state of A

affects states of *B*. But it cannot express, for example, that *A* is part of *B*, as this is not a type of mental representation connectionist networks recognise[1].

Nevertheless, the success of neural networks in pattern-recognition applications became apparent. Their numeric, associative and self-adapting nature enabled them to be employed successfully in applications where symbolic systems did not perform well, such as in the recognition of characters (Carpenter & Grossberg 1992), signal, image and speech processing (Simpson 1990). The main reason for symbolic systems not to perform well in such applications is that the symbolic vocabulary for representing knowledge and reasoning is generally not suitable for pattern-recognition tasks. Due to their intelligible knowledge representation formalisms, and to the transparency with which such systems can treat reasoning, symbolic systems have been addressed to higher-level tasks with more success.

It is noticeable at this stage that the weaknesses and strengths of connectionist and symbolic systems are complementary, as well as their styles of reasoning. Each paradigm mimics well one of two types of thought of the human brain (Kurzweil 1990):

- as mentioned before, neural networks are very well suited for solving pattern-recognition tasks, and in general the tasks involving the five senses in human beings. We do not have a conscious control over such tasks, and thus we cannot explain exactly how we execute them. For instance, we find it difficult to explain how we recognise a face. Another relevant phenomenon is that we find it impossible not to think of elephants when we are asked to do so. Likewise, neural networks can perform these tasks efficiently, but with little or no knowledge of how they achieve the results.

- symbolic systems are well suited for dealing with problems that require some type of logical reasoning. This is the type of reasoning we use to play games and solve

[1]According to Fodor & Pylyshyn (1988a), distributed networks cannot be considered structural representations because no one node on its own in this kind of network represents a concept (only the group of nodes does). Networks that represent concepts with microfeatures do not have structural representations either, because concepts are represented in the form of lists, where it is not possible to establish the direction of relationships, or the elements to which the relationships apply when more than one relationship appears in a list.

problems in a notably sequential manner. In the human brain, these types of thought occur at a more conscious and controlled level. For example, after solving a mathematical problem, we can easily trace our mental processes backwards to explain how we found the solution. In the same way, symbolic systems can perform this type of reasoning with traces that permit a deeper understanding about how the conclusions were reached.

But apart from this suggested cognitive complementarity, other more practical reasons have brought researchers to integrate symbolic and connectionist processing. On the symbolic side, expert system developers have faced several problems related to knowledge acquisition, e.g. the elicitation of knowledge from an expert and translation into one of the knowledge representation formalisms (Barr & Feigenbaum 1986). Until that moment, machine learning researchers had been the main people responsible for the creation of methods to ease the knowledge acquisition phase and to incorporate in expert systems the ability to learn with experience (Michalski, Carbonell & Mitchell 1983). However, the solutions offered also had some drawbacks regarding their ability to perform incremental learning and to maintain the consistency of the acquired knowledge[2]. The idea of using neural networks in the learning process in expert systems therefore seemed to be appealing (Gallant 1988).

With the appearance of systems combining neural and symbolic processes, a number of classification schemes for hybrid architectures were proposed. In Medsker & Bailey (1992), hybrid architectures are classified in accordance with the degree of connection of the symbolic and the neural components. The possible classes are:

- *stand-alone:* the neural and the symbolic components work in an independent fashion. The main reason for the choice of this type of architecture is that the user can compare the performance of the two technologies (ie. neural networks and symbolic system). For instance, an expert system and a neural network can be constructed for the same fault-diagnosis problems. When solving new cases, the user can contrast the performance of the two systems and opt for the results given by one of them, or the

[2]This is mainly related to the difficulties in re-structuring decision trees when new concepts are learned, or keeping the consistency of a rule set when new rules are incorporated.

other. This type of model does not benefit from the integration of the connectionist and the symbolic approaches, as there is no interaction between connectionist and symbolic modules.

- *transformational:* a system conceived with one approach migrates to another approach. For example, the need for heavy numeric processing and strong adaptive capacity justify the conversion of an expert system into a neural network. Transforming a neural network into an expert system, in contrast, may be necessary if better explanation capabilities and richer knowledge representation formalisms are needed. Here there isn't any interaction between the connectionist and symbolic modules, which may restrain this type of architecture from benefiting from the integration between the connectionist and the symbolic approaches.

- *loosely-coupled:* the application is decomposed into separate neural-network and symbolic components, which communicate via data files. For instance, the neural network may be responsible for pattern-recognition tasks, while the expert system may use the information coming from the neural network to solve classification problems, and to explain the reasoning process. Some benefits from the integration of the two approaches are gained in this type of architecture. In the example, some tasks that are handled with difficulty by the expert system are left for the neural network, and vice versa. Loosely-coupled architectures are used mostly to combine existing expert systems and neural networks.

- *tightly-coupled:* the application is decomposed into separate neural-network and symbolic components, which, this time, communicate via resident memory structures. This type of integration enables the establishment of a higher level of interaction between symbolic and connectionist modules. Therefore, more benefits can be gained from the combination of the two approaches.

- *fully-integrated:* the main feature of this type of architecture is that it allows the connectionist and the symbolic modules to share data and knowledge representation structures. The communication between them is made by the dual nature (symbolic/connectionist) of the hybrid representation. For instance, a node of the neural network may also represent a concept of the symbolic system, which makes a clear distinction between symbolic and connectionist components difficult. *Fully-integrated* architectures also try to model human reasoning in a cognitively-plausible way, and by doing so, they benefit both from the numerical nature of the neural networks and from the comprehensible knowledge representation of symbolic systems.

A similar classification scheme has been proposed by Goonatilake & Khebbal (1995) to classify hybrid systems combining different 'intelligent' techniques according to functionality and communication requirements. The possible classes in this scheme are:

- *function-replacing hybrids*, for systems where a function in one technique is performed by another technique. For example, systems that use genetic operators to select weights in a neural network.
- *intercommunicating hybrids*, where independent modules coordinated by a central control perform different functions to generate solutions. For example, an expert system that replaces its symbolic pattern matcher by a neural network.
- *polymorphic hybrids*, where a single processing architecture is used to simulate the functionality of different intelligent techniques. For example, neural networks that can perform some type of symbolic processing.

In Sun (1995a), a yet different classification scheme has been proposed, where four types of integration between neural and symbolic processes are described, according to the type of neural network used and to the degree of connection between the components. The possible classes in this scheme are:

- Architectures in which a *localist*[3] *network is used for symbolic processing*. Each node in the neural network represents a particular concept, and a connection between two nodes represents a liaison between the respective concepts. In such models, the mapping between neural network and symbolic structures is direct.

[3]Although the word *localist* is not common in English, this is the term used by the author when classifying a particular type of hybrid architecture.

- Architectures in which a *distributed network is used for symbolic processing.* The main objective of this type of integration is to make the neural network perform the functional equivalent of symbolic processing in a holistic and functional way.
- Architectures combining *separate symbolic and neural-network modules.* This type of architecture is based on the juxtaposition of a neural network and a symbolic system.
- Architectures that use *neural networks as basic elements in symbolic models*: here, a general symbolic structure is used, but each component of this structure is replaced by a small-scale neural network, in order to enhance the model with more fault-tolerant elements, capable of parallel computation and partial matching.

In this last classification scheme, the architectures described in all but the third item are more symbiotic in that their neural network and symbolic components are intermingled, resulting (as in *polymorphic* and *fully-integrated* architectures) in an organization where the distinction between what is symbolic and what is connectionist can hardly be determined.

The three classification schemes have some similarities, and sometimes overlap. For instance the *fully-integrated* systems of the first scheme are similar to the *polymorphic hybrids* of the second scheme; the *architectures in which a localist network is used for symbolic processing* and the *architectures in which a distributed network is used for symbolic processing* of the second scheme can both be treated as specific cases of *polymorphic hybrids.* In constrast, the architectures defined in some of the schemes cannot be classified in other schemes. For instance, *independent* and *transformational models* cannot be classified by Sun's classification schemes. In Medsker & Bailey's two models, there is no necessary interaction between symbolic and connectionist components, which is not foreseen by Sun's classification schemes.

The architectures that integrate *separate symbolic and neural-network modules* are the ones that relate most to the work developed in this thesis, where the Combinatorial Neural Model has been combined with a separate CBR module. Table 3.1 shows the classification of some of these architectures which are related to our research, according to the three classification schemes presented.

Sytems / Classification	**Medsker & Bailey (1992)**	**Sun (1995a)**	**Goonatilake & Khebal (1995)**
INNATE/QUALMS (Becraft, Lee & Newell 1991)	*loosely-coupled*	*combining separate symbolic and neural-network modules*	*intercommunicating*
Knaus (1992)	*tightly-coupled*	*combining separate symbolic and neural-network modules*	*intercommunicating*
Tirri (1995)	*tightly-coupled*	*combining separate symbolic and neural-network modules*	*function-replacing*
SCRuFFY (Lin & Hendler 1995)	*tightly-coupled*	*combining separate symbolic and neural-network modules*	*intercommunicating*
CONSIDERR (Sun 1995b)	*fully-integrated*	*combining separate symbolic and neural-network modules*	*intercommunicating*
NEXTOOL (Machado & Rocha 1992)	*tightly-coupled*	*combining separate symbolic and neural-network modules*	*intercommunicating*
Yang & Bhargava (1990)	*transformational*	does not apply	*function-replacing*
Kintch (1988)	*tightly-coupled*	*combining separate symbolic and neural-network modules*	*intercommunicating*
LeMICON (Bookman 1995)	*tightly-coupled*	*combining separate symbolic and neural-network modules*	*intercommunicating*
Marker-passing networks	*fully-integrated*	*localist or distributed network used in symbolic processing*	*polymorphic*

Table 3.1: Classifying some hybrid connectionst/symbolic systems

Each of these systems is commented on further in the next section.

3.2 Integrating Neural Networks with Symbolic Systems

In this section we describe several architectures that are related to the research presented in this thesis. These architectures, in most cases, *combine separate symbolic and neural-network components.*

A common approach to the combination of connectionist and symbolic components is the integration of an expert system with a neural network. The system of Becraft, Lee & Newell (1991), for example, combines a model-based expert system (QUALMS) with a neural network (INNATE) to serve as an assistant in the diagnosis of faults in large-scale chemical process plants. The benefits of using a neural network in this diagnostic problem come from the fact that much of the information involved in the diagnostic process is numeric in nature (such as process sensor readings), which is easily handled by neural networks. The expert system is left with the task of handling symbolic information, which is more difficult to treat with neural networks. In the diagnostic process, the network is used as a first-level filter. Its inputs correspond to sensor values or alarm states of the process, and its outputs represent the presence or absence of particular faults. After a fault is identified by the network, the expert system uses its knowledge about the structure and function of the plant to analyse the results, and either confirms the diagnosis or offers alternative solutions. This architecture can be classified as *loosely-coupled,* as it combines a *separate expert system and neural network* that do not share memory and knowledge representation structures. According to Goonatilake & Khebbal's classification, this is an *intercommunicating hybrid* where the neural network performs some initial calculations that are later provided to the expert system for a further analysis.

In Knaus (1992) the main objective has been the representation of the knowledge of experts in neural networks, for a particular problem in architecture. The initial goal of this system was to create an expert system that would give marks to architectural projects, according to some pre-defined criteria. A rule-based system was constructed initially for the application, but the developers were not satisfied with the results obtained, as the kinds of concepts with which the system had to deal (such as aesthetics and efficiency of the architectural project) were difficult to encode in production rules. Thus, an alternative solution was tried, by representing the knowledge of an expert in a neural network. This network was combined with an expert system, the latter being responsible for starting the reasoning process and for finding a preliminar solution for the problem. This solution was then adjusted by the neural network according with the expert's knowledge encoded in the network. This hybrid system has been capable of reproducing the official system of mark attribution for architectural projects. One of its peculiarities is that no learning mechanism is used by the neural network, as the existing criteria for the evaluation of architectural projects should not be modified according to the projects analysed. This architecture can be classified as *tightly-coupled,* as it combines a *separate expert system and neural network* that share memory and knowledge representation structures. According to Goonatilake & Khebal's classification, this is an *intercommunicating hybrid* where the neural network refines the solution initially found by the rule-based system.

Tirri (1995) uses a neural network to support a task performed by a symbolic expert system. In this architecture, the knowledge base of the system is divided into three sub-knowledge bases: a rule-base, a fact-base and a neural network. The rule-based is composed by condition-action rules, each condition being formed by one or more predicates. Each predicate can be connected to a corresponding representation in the neural network. When given a new problem, the inference engine of the expert system performs the normal backward/forward chaining of rules. However, the inference engine may encounter a rule which has a predicate in its condition part for which no value can be determined by the facts in the fact-base. In this case, the system tries to decide on the value of the predicate by activating a corresponding neural network, and using the predicate arguments as parameters for the network. The neural network returns a boolean value that determines whether the predicate was true or false. As this architecture uses a neural network to perform a pattern-matching task, it can be classified as *function-replacing*, *combining separate neural-network and symbolic components.*

Similarly to INNATE/QUALMS and the system developed by Tirri (1995), the SCRuFFY (Lin & Hendler 1995) architecture uses a neural network to support the diagnostic task performed by an expert system. In SCRuFFY, distinct roles are assigned for the connectionist and symbolic components. While a backpropagation neural network is used to classify input signals, a rule-based expert system is used to analyse these signals and make decisions on which actions to take. The reasoning process in SCRuFFY starts with the neural network examining parts of input signals to determine the current state of the system. A temporal pattern matcher is then used to analyse the outputs of the neural network over time, looking for patterns in the consecutive network outputs and trying to provide further evidence for the expert system. The expert system makes control decisions according to the information provided by the pattern matcher, and monitors the effect of the decision from new changes in the input signal. The pattern matcher is therefore the bridge that converts the numeric output produced by the neural network into the symbolic knowledge handled by the expert system. SCRuFFY has been used in the construction of systems for chemical process and welding control, and in the analysis of ballistic signals, all applications related to control problems. This architecture can be classified as *tightly-coupled* and *intercommunicating*, where *separate neural-network and symbolic components* cooperate in the identification of control problems.

The system CONSYDERR (CONnectionist SYstem with Dual-representation for Evidential Robust Reasoning) (Sun 1995b) establishes a much higher level of cooperation between symbolic and connectionist components than that of the systems already reviewed. CONSYDERR is divided into two layers: a symbolic layer where each node represents a concept of the domain, and a microfeature (or connectionist) layer where each node is a fine-grained representation of a concept in the first layer. To express rules, the nodes of the symbolic layer are connected from rule antecedents to rule consequents. All the nodes in the microfeature layer are connected to the concepts in the symbolic layer accordingly. When a node in the symbolic layer is activated, all the nodes in the microfeature layer connected to that concept are subsequently activated for inter-layer action. The activity in the other direction also applies, ie. from microfeatures to concepts. This architecture was applied in natural language understanding, having shown a good ability to solve lexical disambiguation problems. Other types of applications cosidered were day-to-day commonsense reasoning and planning. CONSYDERR can be classified as *fully-integrated* and *intercommunicating*, with *separate connectionist and symbolic components* that have a very high level of interaction.

The system NEXTOOL (Machado et al. 1991; Machado & Rocha 1992) has a different approach from the ones presented so far in that it uses a symbolic system (semantic network) with the main purpose of representing declarative knowledge about the domain, as well as some procedural knowledge which has a correspondence with the knowledge stored in the neural network. In the approaches reviewed, the focus has been on the use of the symbolic component to represent mainly causal relationships between concepts (e.g. procedural knowledge represented in rules). Little or no attention has been given to the representation of structural aspects of the domain knowledge, which can be achieved with the use of mechanisms such as semantic networks and frames. The neural network used by NEXTOOL is the Combinatorial Neural Model (CNM), which stores knowledge complementary to that of the semantic network. While the semantic network represents declarative knowledge of the domain, the CNM is responsible for learning and refining classification patterns, in addition to carrying out the reasoning in the system. The integration of the symbolic and the connectionist component in this architecture is achieved by influence links that express causal relationships in the semantic network and that have corresponding connections in the neural network. Thus, the symbolic influence links and the neural network can be also seen as representing the same classification problem. The semantic network is used to guide the consultation process, collecting input information for the neural network. The network propagates the activation of the input nodes to the output nodes (classes or diagnoses). In case the evidence provided is not sufficient for the neural network to reach a conclusion, the semantic network guides the questioning of the user for further information. This architecture has been employed successfully in the development of a system for the diagnosis of renal syndromes: uremia, nephritis, calculosis and hypertension (Machado et al. 1991). The model of integration of neural and symbolic processes used by NEXTOOL has had a great deal of influence on the work presented in this thesis, and will be treated in more detail in the next chapter. It can be classified as *tightly-coupled, intercommunicating*, where *separate connectionist and symbolic components are integrated.*

Yang & Bhargava (1990) have had a different approach to combining neural networks with symbolic systems. They designed a *transformational* model that uses a neural network to implement a rule-based expert system. The neural network was used initially to learn classificational knowledge, and it was later converted into a rule-based system. The rules handled by the system considered the following types of correlation between attributes:

- logical concurrence: when input F_i is highly correlated with output D_j;
- negative concurrence: when there is a strong negative correlation between an input F_i and an output D_j.
- disjunction: output Dj occurs whenever either input F_i or F_k is observed;
- exclusive disjunction: output Dj occurs whenever either input F_i or F_k occurs, but not both;
- sufficient implication: if all of inputs F_{i+1}, $F_{i+2,...,}$ F_{i+k} occur, then Dj occurs too;
- necessary implication: if input F_i does not occur, then Dj does not occur either.

Although no interaction in the reasoning process exists between the connectionist and the symbolic components, this architecture is able to take advantage of their integration: while neural networks are used to learn and refine knowledge, a rule-like representation is available for consultation and explanation purposes. This model could be also classified as *function-replacing*, as induction algorithms, more commonly used to build rule-based systems, have been replaced here by a neural network.

Hybrid connectionist/symbolic systems are also becoming more popular in language-understanding applications. The model in Kintch (1988), for instance, used a symbolic production system to build symbolic representations of the alternative interpretations of text and to construct a localist network in which the different interpretations competed. While the localist network was used for disambiguation purposes, the symbolic production system allowed the model to perform the rule-firing and inferencing that is difficult for connectionist models. More recently, LeMICON (Bookman 1995) has made use of connectionist and symbolic techniques to construct plausible interpretations of text. The system's architecture supports both structured and non-structured representations[4]. While structured representations

[4]In structured representations each node of the network represents a concept, and a connection between two nodes represents a relationship between concepts. In non-structured representations, a single concept

provide intelligibility and comprehensibility, non-structured representations enable similarities to be found between texts that have no explicit connections.

Another area of research that is related to connectionist/symbolic processing is that of marker-passing networks. These networks are different from neural-network models in that their units do not have numeric activation functions. Instead, their units spread their activation by propagating symbolic markers through the network. This feature makes marker-passing networks to be known as the most symbolic of the connectionist models. Because of this symbolic character, marker-passing networks units do not have the graded level of activation seen in other neural-network models, which gives marker-passing networks an 'all-or-nothing' character[5]. This is one of the main drawbacks of the approach. Furthermore, marker-passing networks do not possess the learning capabilities observed in other connectionist models (Lange 1992). But despite these difficulties, marker-passing networks have been used successfully in areas such as planning (Hendler 1988), and natural language understanding (Charniak 1986).

3.3 Difficulties with Hybrid Symbolic/Connectionist Approaches

It is possible to observe from the description of the systems presented in the previous subsection that two main reasons have led researchers to the development of hybrid architectures combining connectionist and symbolic approaches. The first is related to an engineering problem: in certain circumstances no one single approach can provide the appropriate means for knowledge representation and reasoning in a given domain. Apart from *fully-integrated* and *polymorphic hybrids*, the other types of architecture are usually designed for this reason. For instance, the system NEXTOOL (*tightly-coupled, intercommunicating*)

---

can be represented through several interconnected nodes. This last form of representation is notedly less intelligible.

[5]In Hendler (1989), a marker-passing network is expanded to represent concepts with microfeatures and to contain numeric activation functions that represent the activation strength of a unit. Therefore, units that receive strong activation become part of a marker path, whereas units that do not receive enough activation cannot support further marker propagation.

employed a symbolic semantic network to represent domain knowledge, and a neural network to learn and perform classification tasks. INNATE/QUALMS (loosely-coupled, intercommunicating) opted for using a neural network to process signals (which can be done efficiently by connectionist systems), and for using an expert system to analyse these signals and make decisions.

The second reason for combining symbolic and connectionist approaches is related not only to the complementarity of their nature, but also to the possibility of modelling cognitive processes in a plausible way by using both approaches. The architecture of CONSYDERR, for example, provides a flexible knowledge representation mechanism which can be used in a variety of tasks. Additionally, by having a two-layered knowledge representation scheme (one layer representing microfeatures, the other layer representing more complex symbolic concepts), the architecture proposes a plausible model for learning and reasoning. By having the connectionist microfeature layer connected to the symbolic concept layer, this architecture also provides a solution for one of the best-known problems of neural networks, ie. the representation of the numerical knowledge of neural networks in a comprehensible symbolic fashion.

The system NEXTOOL also proposed a solution to this problem, by connecting the nodes of the neural network to *influence links* of a semantic network. However, neither NEXTOOL nor CONSYDERR are able to provide a symbolic representation, in a graded scale, of the importance of feature/value pairs in the identification of a class (or diagnosis). This problem has been tackled by HYCONES (Reategui & Leao 1993), which is presented in the next chapter, and later by NN-CBR, presented in chapter 5.

Another problem on which hybrid symbolic/connectionist systems do not focus is that of referring to specific previous experiences when solving a problem and proposing a solution. This has been considered as an intuitive problem-solving approach (Kolodner 1993), as well as an effective way of supporting results achieved (Georgin et al. 1994). Our idea in this thesis has been to incorporate, in a single architecture, the capacity of referring to previous experiences, the knowledge representation facilities of symbolic systems, and the learning capability of neural networks.

The next chapter presents some previous research that has led eventually to the results obtained in the work presented in this thesis. In this previous research, many experiments were carried out. Some of these experiments proposed solutions for the integration of symbolic and connectionist approaches, and tackled problems such as the symbolic representation of neural-network knowledge.

Chapter 4

Knowledge Graphs, CNM, HYCONES and other Experiments

The work reported in this thesis represents a natural next development of some previous research conducted by Dr. Beatriz de Farial Leao, at the Cardiology Institute (Porto Alegre, RS - Brazil) and myself, at Universidade Federal do Rio Grande do Sul (Porto Alegre, RS - Brazil). This chapter summarises the previous research and introduces some topics that are important for the understanding of the system presented later: for example, knowledge graphs, the neural network CNM, the systems NEXTOOL and HYCONES. We also describe in this chapter other more recent experiments carried out at University College London. Three systems have been developed in these experiments, each of them based on a different architecture and applied to a particular domain, namely classification of credit card transactions, identification of the profile of patients in evaluation for heart transplants, and diagnosis of Congenital Heart Diseases.

4.1 Introduction

The knowledge representation formalism of knowledge graphs has been conceived with the main goal of providing a means for the represention and combination of knowledge elicited from multiple experts. The need for such a mechanism to elicit and represent knowledge can be exemplified in the environment of a large company (e.g. a bank). It may be interesting for such a company to build a corporate knowledge base using the expertise of its more prominent specialists in a particular problem domain (e.g. credit aproval). In order to build this corporate knowledge base, the elicitation and normalising of the knowledge of the company's specialists is required. However, if the task of eliciting knowledge from one expert is already complex, the difficulty becomes much higher when a large number of experts is concerned.

The knowledge acquisition methodology of *knowledge graphs* provides the means for eliciting and representing the knowledge of multiple experts (Machado, Rocha & Leao 1990). Several experiments in this direction have been carried out. For instance, Greco & Rocha (1988) showed the soundness of the knowledge graph approach for the analysis and description of text understanding in a population, finding a significant correlation between this understanding and the electrical brain activity, measured through the electro-encephalogram. Theoto et al. (1987) have shown that this approach can be used to characterise the different understandings that a same text can have in populations with different backgrounds of knowledge about a specific subject. Later, Leao & Rocha (1990) used knowledge graphs to highlight the observation that the knowledge representation models built by experts in congenital heart diseases are smaller and less complex than those built by non-expert cardiologists.

In addition to having been used in all these experiments, the knowledge graphs have also been the source of inspiration for the development of the Combinatorial Neural Model (CNM) (Machado & Rocha 1990), which is the neural network used in our research.

The next section presents the knowledge representation formalism of knowledge graphs and the methodology used in order to elicit the graphs. Section 4.3 describes the CNM, a neural network which has a very similar structure to that of the graphs for representing classification knowledge. Section 4.4 describes the system NEXTOOL, which combined the CNM with semantic networks in the development of hybrid expert systems. Section 4.5 describes HYCONES (Reategui & Leao 1993; Leao & Reategui 1993), a hybrid symbolic/connectionist system that has been based on NEXTOOL and which set the grounds for the development of the architecture presented in this thesis. Section 4.6 presents other more recent experiments which have been carried out at University College London.

4.2 Knowledge Graphs

A Knowledge Graphs (KG) is defined as a directed AND/OR acyclic graph used to represent the knowledge of an expert (or a group of experts) for a diagnostic hypothesis. Figure 4.1 shows the basic structure of the KG.

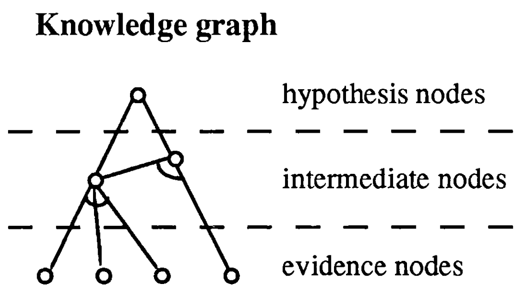


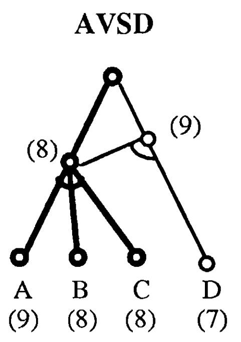
Figure 4.1: The basic structure of the knowledge graph

Three types of nodes can be identified in the graphs:

- *hypothesis nodes:* representing the diagnostic hypotheses considered in the graph;
- *evidence nodes:* representing symptoms, tests results or any other information that supports the diagnostic hypothesis. They appear in the graph in their order of importance, from left to right.
- *intermediate nodes:* representing different groupings of evidence used by an expert for reasoning about a problem.

Figure 4.2 gives an example of the use of a KG for the representation of diagnostic knowledge for the diagnosis of *Atrioventricular Septal Defect* (AVSD), a common congenital heart disease.

Knowledge graph



AVSD = Atrioventricular septal defect
A = Left anterior hemi blockage
B = Down's syndrome
C = Cardiomegaly on CXR
D = Increased pulmonary flow on CXR

Figure 4.2: Knowledge graph for the diagnosis of AVSD

The graph shows that there are two different trees that lead to the diagnosis. The first tree consists of the three leftmost nodes of the graph (A, B and C) connected by a logical AND in an *intermediate node*. The second tree consists of the fourth node of the graph (D), connected by a logical AND to the first tree. The two different trees can be described as two different alternatives to reach the diagnosis, which means that they become connected by a logical OR.

The knowledge acquisition method for the elicitation of knowledge graphs starts with a series of interviews with an expert, where he or she has to determine the set of problems to be solved, as well as the evidence that influences the identification of each problem. For instance, in a medical context, the expert would have to determine the diseases that the system would try to diagnose, as well as the symptoms and the result of clinical examinations that amount to determining-factors for the identification of each disease. Then, for each of the problems considered, the expert would have to sort the pieces of evidence selected, according to their order of importance. These items are placed in the *evidence nodes* layer, while the possible solutions for the problems (diagnoses) are placed within the *hypothesis nodes* layer. In the next phase, the *evidence nodes* that have some particular degree of importance when seen together are connected in an *intermediate node*. The *intermediate nodes* are then connected to the *hypothesis nodes*. In the last phase of the construction of the graphs, an importance degree, in a 1 to 10 scale, is assigned to each of the nodes. The detailed knowledge acquisition method for the elicitation of knowledge graphs can be found in Appendix A.

4.3 The Combinatorial Neural Model

The knowledge acquisition methodology of knowledge graphs inspired the development of the Combinatorial Neural Model. The structure of the CNM is therefore very similar to that of the graphs.

Table 4.1 (adapted from Denis & Machado (1991)) summarises the main features of the CNM and contrasts them with other well-known connectionist models.

model/ features	**Network properties**	**Learning**	**Example of applications**
Perceptron (Rosemblatt 1962)	Acyclic network with 2 layers	Reinforcement: supervised learning where the connection weights are rewarded when the system performs correctly and punished when it does not	character recognition
Backpropagation (Rumelhart, Hinton & McClelland 1986)	Acyclic network with at least 3 layers	Error correction: supervised learning that adjusts the connection weights proportionately to the difference between the expected output and the actual values obtained in the output layer	speech processing, adaptive control of robot arm movements
ART (Grossberg 1976)	Cyclic network with 3 layers	Competitive or cooperative: non-supervised learning where connections can inhibit neighbouring nodes (competitive) or excite them (cooperative)	radar and sonar signal recognition, image processing

Kohonen (1988)	Cyclic network with 2 layers	Learning based on randomly-connected systems: non-supervised learning that supports the theory that the human brain is similar to a network randomly connected when analysed from a macroscopic level	speech recognition, learning the probability distribution of data
Hopfield (1982)	Cyclic network with 1 layer	The connections and their weights are pre-determined (ie. before the network is activated)	recognition of data or complete images from fragments, associative memory
CNM (Machado & Rocha 1990)	Acyclic network with 3 or more layers	Supervised learning similar to backpropagation (error correction) where punishments and rewards are computed for each connection of the network. The actual connection weights are calculated through the normalisation of the punishment and reward accumulators	classification, diagnosis of medical diseases

Table 4.1: Comparison of neural network models

The Perceptron (Rosemblatt 1962) has been the first connectionist model developed. This network aroused a lot of interest at the time, for its ability to recognise linearly separable patterns. However, the model is not appropriate for classes that are not linearly separable, as in the problem of the exclusive-OR (XOR). This problem has been used by Minsky and Papert (1969) to illustrate the deficiencies of the Perceptron, which contributed to the stagnation in neural network research for about a decade.

The backpropagation algorithm (Rumelhart, Hinton & McClelland 1986) has been designed to train multi-layer Perceptrons (networks with one of more intermediate layers between the input and output layers). This algorithm has been tested in different problems, such as the exclusive-OR, in problems of visual pattern-recognition and speech processing. In most circumstances, the backpropagation found good solutions for the problems, although the algorithm, sometimes, gives a weight configuration for the network that would correspond to a local mimimum of the error-function.

Grossberg (1976) has developed the network ART, which uses a non-supervised learning mechanism to form information clusters. In this network the connections between nodes are bi-directional. The bottom-up connections try to classify the input stimuli, while the top-down connections try to learn how to cluster the stimuli (ie. how to form classes). The learning process selects the first input presented to the network as an initial cluster. The next input is compared with the first example. They are grouped in the same cluster if the distance between the two examples is smaller than a given threshold. Otherwise, the new example will start another cluster. This process is repeated for all the input examples available. The number of clusters grows in accordance with the threshold function selected, as well as with the distance metric used to compare the training examples.

At the beginning of the 80s, Kohonen (1988) proposed a network that was based on some theoretical work on the organisation and functioning of the human brain. A spatial arrangement of processing units was used in the design of the network, which enabled the structured representation of input patterns. A self-organisation mechanism operates as a clustering algorithm. When some stimuli form a cluster in the input area of the network, the algorithm maps the patterns of the cluster to a specific area in the network. The algorithm arranges these areas so as to capture the general topology of patterns observed in the input clusters. Kohonen has shown that this algorithm could be used for problems of speech recognition, where the network could learn a phonological map with distances between units that were proportional to vectorial-distances shown as input examples.

The network built by Hopfield (1982) had as a main goal to serve as a self-associative memory, or to be used in optimisation-problems. This network is more appropriate for problems that can be modelled through a binary-represention, such as black and white images where the input elements can be represented by the values of each image dot (eg. 0 = white, 1 = black). When used as a memory accessible by content, this network presents a problem. Despite the patterns shown and stored by the neural network, it can converge to a representation which is different from the examples presented to it. This may result in a situation where the network is not capable of finding an existing pattern. Moreover, if a training example shares too many representation bits with another example, the network may converge to a representation uniquely for this second example.

The CNM network has been conceived more recently (Machado & Rocha 1990). The main reason for the development of this neural network model has been to provide a computerised method that would use the same reasoning model of the knowledge graphs. Moreover, by having a neural network with a similar structure to that of the knowledge graphs, machine learning and knowledge engineering technologies could join their efforts in the development of intelligent systems. For instance, graphs elicited from experts could have the importance degree of their nodes adjusted by the neural network. This could be achieved easily by mapping the importance degree of the graphs into connection weights of the CNM network, and using the network's refinement algorithm to adjust these weights according to a set of training examples.

The CNM has a feedforward topology with three layers[1], where:

- output layer: contains nodes that represent the different diagnostic hypotheses;
- input layer: contains nodes that represent evidence such as symptoms, test results or any other information that supports the diagnostic hypothesis.
- intermediate layer: specifies different combinations of evidence that can lead to a particular diagnostic hypothesis.

The main difference between the CNM and the other neural network models presented in table 4.1 relies on its different structure and learning algorithms, which are based on the reasoning model of the knowledge graphs. The CNM has been used in a variety of tasks, including toy problems (Denis & Machado 1991) and real-world applications. These were usually higher-level tasks, such as medical diagnoses (e.g. renal syndromes (Machado & Rocha 1992) and congenital heart diseases (Leao & Reategui 1993)) or other types of classification (e.g. credit card transactions (Reategui & Campbell 1994)). Other neural

[1]Although the CNM can assume a configuration of three layers or more, usually three layers are sufficient to represent the problems addressed.

network models have also been developed to address such higher-level problems. These models have been classified as connectionist expert systems, and were employed in tasks such as the diagnosis and treatment of sarcophagus diseases (SEC (Gallant 1988)), identification of film titles (RUBICON (Samad 1988)), diagnosis of fever and identification of precious stones (FUZZNET (Romaniuk 1989a)).

The main difference between the CNM and these other models remains once again in the CNM's learning algorithm and in the activation functions of its cells (fuzzy-and, fuzzy-or). Another peculiarity of the CNM in relation to other neural network models is that it can keep pathognomonic pathways, ie. the ones that have rewards but no punishments, even if low values appear in the reward accumulators. This feature endows the CNM with the ability of dealing with two concepts often used in medical domains, ie. specificity and sensitivity (Owens et al. 1990). Besides reinforcing frequently-successful pathways, the CNM is capable of identifying features that are specific to a class (even if not too frequent). These features are given a significant weight by the CNM[2]. The Combinatorial Neural Model will be explained in detail in chapter 5.

4.4 The System NEXTOOL

NEXTOOL (Machado et al. 1991; Machado & Rocha 1992) is a hybrid tool for the development of expert systems which makes use of a semantic network to represent the domain knowledge, and the CNM network to solve classification problems. NEXTOOL's semantic network is devided into two parts:

- *An Intensional Semantic Network:* involving only the classes of concepts (objects) and a set of primitive relationships.
- *An Extensional Semantic Network:* where the classes of objects are instantiated.

This organisation of the semantic network provides a clear differentiation between expressions at the conceptual level and the statements at the extensional level. Figure 4.3

[2]Another example of a neural network that tries to work with specificity and sensitivity concepts is SMART (Guazzelli & Leao 1994), a modified version of the ART model based on the CNM.

shows a fragment of a semantic network representing knowledge related to the diagnosis of *Appendicitis*. Four important classes are defined in the example:

- *Hypotheses:* represent the categories (or classes) of the classification problems;
- *Attributes:* represent features that provide information about the subject under analysis;
- *Evidence:* represents the possible outcomes of attributes;
- *Procedures:* represent the tasks that are executed to measure one or more attributes.

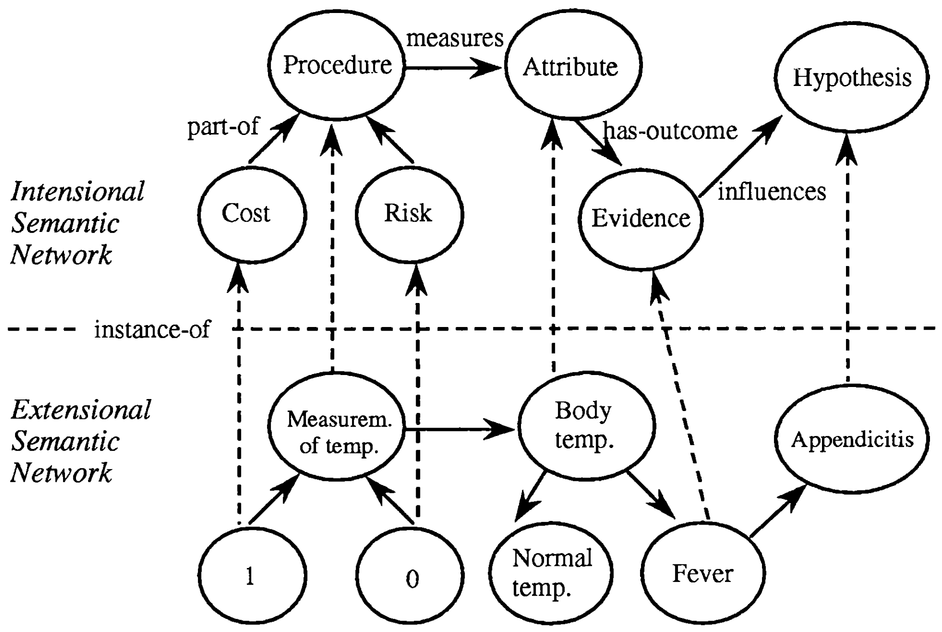


Figure 4.3: NEXTOOL's semantic network divided into *Intensional* and *Extensional levels*

In the example, a *Procedure* for determining the value of an *Attribute* has an associated *Cost* and *Risk.* Each *Attribute*, when assuming certain values, may represent a piece of *Evidence* which can be used to indicate a particular diagnostic *hypothesis*. The *Extensional Semantic Network* for this same example shows that the *Measurement of temperature* of a person has a given *Cost* (1), but no *Risk.* A *Measurement of temperature* indicating *Fever* represents a piece of *Evidence* for the identification of *Appendicitis*.

The integration of the symbolic and the connectionist component in this architecture is achieved by the *influence* links of the semantic network, which have corresponding connections in the neural network. The subset of the semantic network composed by the relationship *influences* and its associated objects in the *Extensional Semantic Network* are called *Influence Network.* Every object belonging to the *Influence Network* will have a neuron associated in the neural network, whose activation will represent its possibility degree. For instance, 'the patient has appendicitis with possibility 0.7' means that the neuron associated with the object *Appendicitis* of the semantic network presents an activation equal to 0.7.

The semantic network is used in NEXTOOL mainly to guide the consultation process and to collect input information for the CNM network. The neural network is the actual mechanism in the hybrid scheme responsible for determining the solutions for the diagnostic (or classification) problems. The network operates by propagating the activation of the input nodes to the output nodes (classes or diagnoses). In case the evidence provided is not sufficient for the neural network to reach a conclusion, the semantic network guides the questioning of the user for further information.

The model of integration of neural and symbolic processes used by NEXTOOL has had a great deal of influence on the design of HYCONES, a system for the diagnosis heart diseases. The system HYCONES is presented in the next subsection.

4.5 The System HYCONES

HYCONES is a HYbrid CONnectionist Expert System developed for the diagnosis of Congenital Heart Diseases (CHD) (Reategui & Leao 1993; Leao & Reategui 1993). The system was conceived with the main purpose of trying to improve some of the knowledge-modelling facilities of NEXTOOL, by replacing NEXTOOL's semantic network with a mechanism of frames. HYCONES' architecture was also intended to employ the symbolic component (frames) for representing both declarative and procedural knowledge of the domain, and the connectionist component (CNM) to learn classification patterns and carry out the reasoning in the system.

Figure 4.4 shows the architecture of HYCONES.

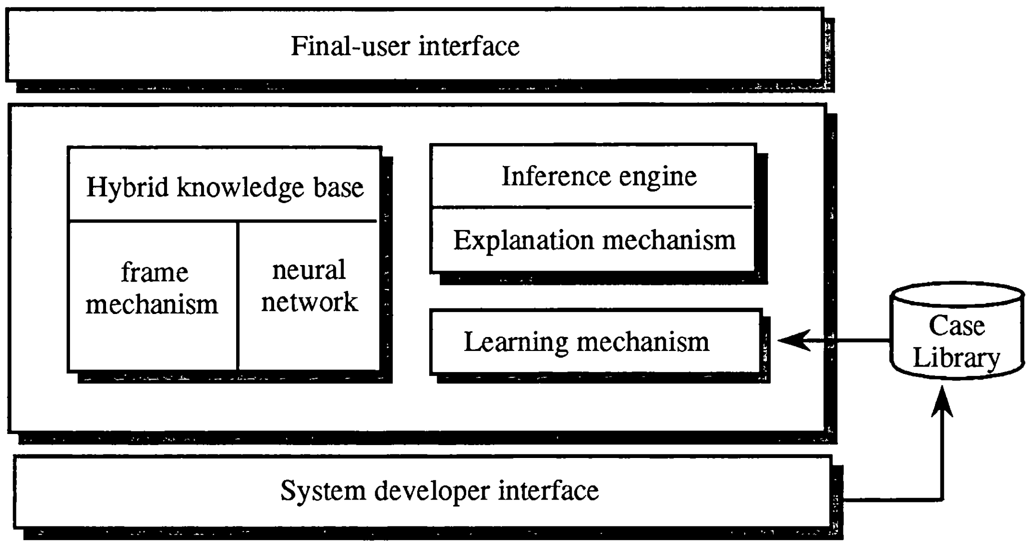


Figure 4.4: The architecture of HYCONES

The *Hybrid knowledge base* consists of a combination of a frame mechanism with the CNM neural network. The frame mechanism provides the system with facilities for the representation of the domain knowledge according to the four classical abstraction concepts: generalization, classification, aggregation, and association (Hull & King 1987). Furthermore, the symbolic knowledge stored in the frame system is used by the *Explanation mechanism* in order to build explanations for the reasoning carried out. The neural network , in contrast, is used by the *Inference engine* to solve new problems, and by the *Learning mechanism* to acquire or refine knowledge about classification problems. In this architecture, the *Case library* serves the exclusive purpose of providing training examples for the *Learning mechanism.*

Figure 4.5 gives an example of the description of domain knowledge through the use of the abstraction concepts.

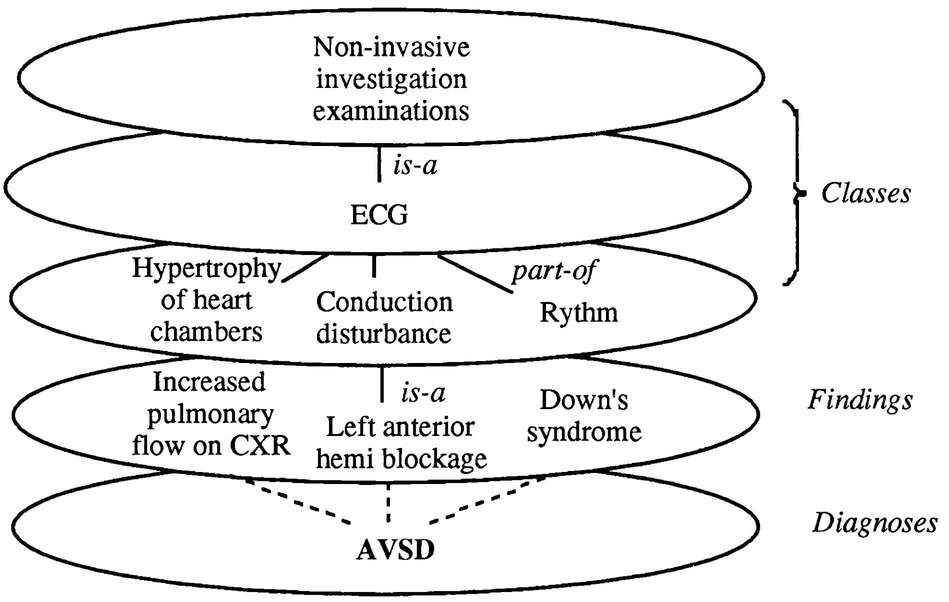

Non-invasive investigation examinations
is-a
ECG
Classes
Hypertrophy of heart chambers
Conduction disturbance
part-of
Rythm
Increased pulmonary flow on CXR
is-a
Left anterior hemi blockage
Down's syndrome
Findings
AVSD
Diagnoses

Figure 4.5: Representation of the domain knowledge in HYCONES

Three main levels can be observed in figure 4.5: a level of classes, a level of findings and a level of diagnosis. The levels of classes and findings are used to describe the declarative aspects of the domain knowledge, represented in figure 4.5 as the first four layers, from top to bottom. The finding *Left anterior hemi block* is connected to the class *Conduction Disturbance* by an is-a arc (classification). This class is then connected to the Electrocardiogram (ECG) class by a part-of arc, indicating that *Conduction disturbance* is one of the components of an ECG examination. The class ECG is connected to a more generic class - *Non-invasive investigation examinations* - by another is-a arc, depicting the use of a generalisation concept. The connection between the levels of findings and diagnoses is acomplished through links that indicate the influence of a finding in the identification of a diagnosis. These symbolic links also represent the connections of the neural network from evidence nodes (*input layer*) to hypothesis nodes (*output layer*). The integration of the symbolic and the connectionist components of this architecture is therefore similar to that of NEXTOOL, where influence links of a semantic network expressed the causal relationships also observed in the neural network. However, HYCONES ranks these influence links into four degrees of importance, and presents them in the form of *diagnosis descriptors* where:

- *triggers:* reference a finding that, when present, indicates the diagnosis as a potential solution to the problem;
- *primary findings:* reference a list of findings that can assure the diagnosis identification;
- *secondary findings:* reference a list of findings that can increase the confidence in the diagnosis;
- *negative findings:* reference a list of findings that can exclude a diagnosis from the set of possible ones.

A particular method has been used to map the knowledge stored in the CNM neural networks into the *diagnosis descriptors.* This method was based on the computation of the sensitivity and sensibility degrees of the findings observed in a set of training examples (Owens et al. 1990). The guidelines for building a *diagnosis descriptor* are the following:

- *trigger:* the most specific of the group of findings which is frequently observed for the diagnosis.
- *primary findings:* findings frequently observed for the diagnosis, showing a sensitivity degree higher than a threshold previously set;
- *secondary findings:* findings that have a high sensitivity degree, but not as high as those referenced by *primary findings.*

Because of the difficulties in identifying negative evidence, no automated method was used in the selection of *negative findings.* Instead, these findings would be selected explicitly by an expert. They would be represented as connections with negative weights in the neural network, as well as in the *negative findings* attribute of the *diagnosis descriptors.*

In HYCONES the inference process starts after the collection of findings from the environment, which is guided by the symbolic component that stores the frame hierarchy (item *4a* in chapter 2 - *general knowledge supporting CBR*). These findings would correspond to the inputs of some of the neural networks. Based on the evidence given, the networks are activated, indicating some *diagnosis descriptors* as the potential solutions to the problem. Attached to each of these *diagnosis descriptors* there is an evidential factor, obtained from the respectively-triggered neural network. The *diagnosis descriptor* showing the highest evidential factor will be the final solution. However, this value must be higher than a previously-defined threshold. Whenever the final evidential factor is below this threshold, the system requests further information from the user. This request is based on the list of essential and complementary findings of the triggered *diagnosis descriptors*. Additional evidence is then supplied to the neural network. At the end of this cycle, the system presents the final conclusion, along with an explanation of the reasoning process. This explanation is obtained through the analysis of the observed findings for the case, which correspond to the *trigger*, the *primary* and the *secondary findings* of the *diagnosis descriptor* carrying the highest evidential factor. HYCONES' main contribution has been the definition of the mechanism for integrating the formalism of frames with the CNM network.

4.6 Other Prior Experiments

By using the four abstraction concepts, the system HYCONES attempted to improve on NEXTOOL's facilities for the representation of domain knowledge. Moreover, HYCONES has introduced a mechanism that enabled the symbolic representation of the importance of feature/value pairs in the identification of a class (Reategui & Leao 1993). However, HYCONES could still not refer to specific previous experiences when solving a problem and proposing a solution.

Some later experiments that focused on this problem, and which led to the development of NN-CBR, have included:

- *CCard*: classifying credit card transactions (Reategui & Campbell 1995);
- *ChartD2*: diagnosing congenital heart diseases (Reategui, Campbell & Leao 1996);
- *Profile*: identifying the profile of patients in evaluation of suitability for heart transplants (Reategui, Campbell & Borghetti 1995).

The next subsection describes in detail the system *Ccard.* The other two systems are summarised in subsection 4.6.2, where a comparison of all the experiments is presented.

4.6.1 Classifying Credit Card Transactions

One of the major problems related to credit card theft is that some transactions which appear after the theft nevertheless have to be accepted and payed for by the credit card companies, as they were carried out by the credit card owner and not by a thief. The credit card companies do not have an efficient way of discriminating between these transactions and fraudulent ones, leading to the loss of large amounts of money every year.

A hybrid architecture integrating a CBR system with a neural network has been devised to solve this problem. The CBR system keeps track of all the transactions carried out with a particular card after the reported theft. When a new transaction appears, the CBR system looks for best matches in the set of previous transactions. The neural network learns general patterns of use and misuse of credit cards through the analysis of old cases, and uses this knowledge to decide when to grant or deny authorization to transactions.

This achitecture made use of a *Central Control*, ie. the CBR system and the neural network are coordinated by a central device which requests services from one or both of the mechanisms. This central device uses the answers coming from the neural-network and the CBR components to determine a final result.

The system designed to solve the credit card problem consists of 5 main components: *case library*, the *knowledge base*, the *neural network*, the *CBR system* and the *Central Control.* The following subsections detail each of these components.

4.6.1.1 The case library

A case is seen here as a collection of transactions for a particular account. The role of the *case library* is therefore to provide the cases for the CBR system and for the training of the neural network.

All the transactions that happened after the reported theft, in addition to a few transactions that happened before the theft, are kept for each case of credit card theft. A typical transaction record is presented in figure 4.6.

Account: 715389819	
Theft date: 28/10/91	**Amount:** *84.91*
Transaction date: 22/01/92	**Type of company:** *5311*
Company: *Galeries Lafayette*	**Transaction type:** *FNG*
Location: *Paris*	**Status:** *To be denied authorisation*

Figure 4.6: A typical record for a credit card transaction

The field credit card *Account* contains the number of the account of the stolen card, and *Theft Date* refers to the date on which the credit card was stolen. The meaning of the next three fields are self-evident. The field *Type of Company* refers to categories in which companies are classified, e.g. supermarkets, restaurants, petrol stations, etc. In the example given in figure 4.5, the code *5311* refers to department stores. The field *Amount* shows the amount in sterling to be debited in that credit card account. The field *Transaction Type* contains a three digit code which breaks down the transactions into categories describing such things as whether the transaction was carried out in Britain or abroad and whether the purchase was for consumables or non-consumables. The field *Status* refers to one of the two categories in which a transaction can be classified, i.e. *to be granted* or *to be denied authorization.*

4.6.1.2 The Knowledge Base

The *Knowledge Base* stores all the domain knowledge used to process the data coming from the outside world and transform it into appropriate input to the CBR system and the neural network. The domain knowledge permits representation in an hierarchical frame scheme which makes use of the four basic abstraction concepts: generalization, classification, association and aggregation (Hull & King 1987). Figure 4.7 presents three excerpts of the frame hierarchy for the credit card system depicting the use of the abstraction concepts. Besides providing flexible constructs to model the domain knowledge, the abstraction concepts can offer additional pieces of evidence for the inference mechanism. For example:

- the set of transactions called *Company Categories* has been defined in order to inter-relate different *Company types*. For example, the *Company Types*: *Children clothes*, *Men clothes* and *Sports Clothes* can be associated in a group called *Clothing*. Therefore, transactions carried out in companies with different types, which belong to the same category, present a good degree of similarity.
- the aggregation *Country* is composed of counties, which are composed of cities, etc. Transactions carried out in different locations, but in the same city have a good similarity degree, while transactions in different countries, for instance, present a much lower similarity degree.

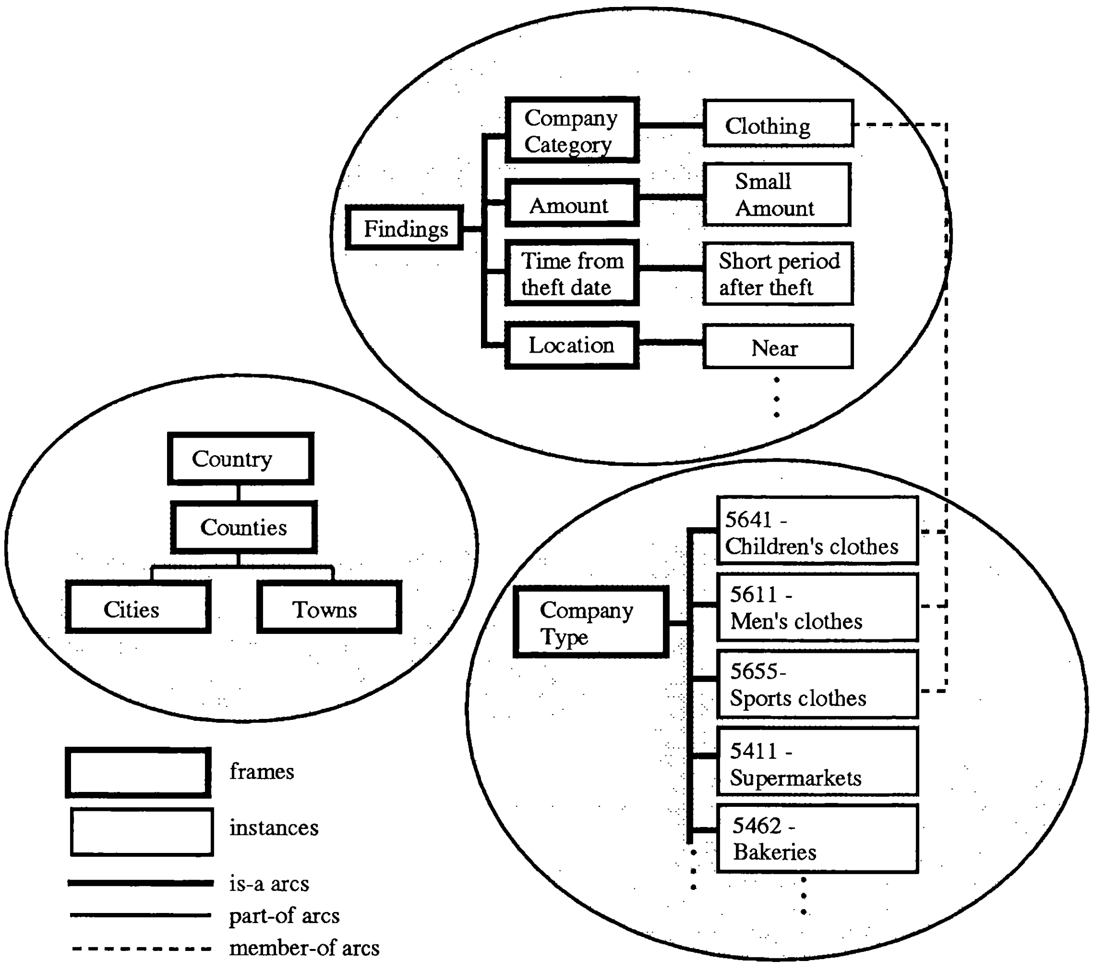


Figure 4.7: Examples of the use of the abstraction concepts in the frame hierarchy

4.6.1.3 The Neural Network

We have used the Combinatorial Neural Model (CNM) as our choice of neural network (presented in section 4.3). The CNM is employed here to recognise general patterns of behaviour for the use and the misuse of credit cards, and use this knowledge in the classification of new transactions. After a learning period, the neural network is able to recognise, for instance, that *donations* or *magazine subscriptions* are usually classified with the status *to be granted*, and transactions envolving large amounts of money are usually classified with the status *to be denied authorization.* While the neural network stores general knowledge about use and misuse of credit cards, the CBR system is left with the task of learning how each customer used his or her credit card, and detecting discrepancies in their habitual way of using their cards.

4.6.1.4 The CBR system

The idea behind CBR is to emphasise the use of concrete case instances in problem-solving. The CBR system takes the transactions carried out with a particular credit card as a source of knowledge to be used in the classification of subsequent transactions for the same card. When classifying a new transaction, the CBR system assumes that if there is an old transaction that matches the new one with a good similarity degree, they must both be classified in the same way.

As the number of transactions for each case of credit card theft is not too large, the transactions for each case can be kept in the *Case Base* in a flat structure. Thus, a best match for an incoming transaction can be found by searching serially the set of transactions stored for that same case of card theft. The CBR system determines the degree of match between two attributes by measuring the distance between them in a qualitative scale. When the values are within the same qualitative region, they present a good degree of match. For instance, if the interval *Small* for the attribute *Amount of Money* were defined as [0,18], the values 12.00 and 16.00 would be considered small, and therefore would match perfectly. However, having fixed boundaries for the intervals can represent a problem. For example, if the next interval *Average* were defined as [19,50], there would be no similarity between the amounts 18 and 19, while the amounts 19 and 20 would match perfectly. Fuzzy overlapping boundaries for each qualitative region have been defined to minimise this problem. Figure 4.8 depicts the definition of the overlapping intervals *Small* and *Average*.

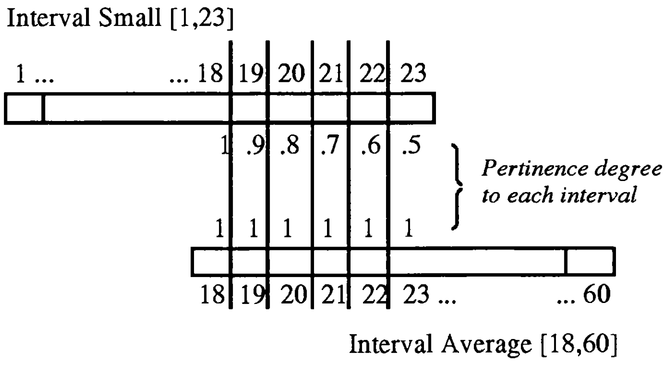


Figure 4.8: Example of fuzzy overlapping intervals

The CBR system also takes into account how important an item of evidence is in the classification process when calculating the degree of match between attributes. For example, the attribute *Location* is more important than the attribute *Company Type*, as *Location* might give the reasoner a better indication of whether the owner or the thief used the credit card for a particular transaction. The importance of each attribute was determined through the analysis of the human specialists' strategy to classify transactions. The similatity between two transactions is thus computed through the formula:

$$Sim\,(T_a, T_b) = \sum_{i=1,n} (Importance(At_i) \times Match(Val_a(At_i), Val_b(At_i)))$$

where T_a and T_b are the transactions being compared, n is the number of attributes defined for a transaction, At_i is the i^{th} attribute, $Val_a(At_i)$ is the value of the attribute At_i for transaction T_a, $Val_b(At_i)$ is the value of attribute At_i for the transaction T_b, *Importance* (At_i) returns the importance of the attribute At_i and $Match(Val_a(At_i), Val_b(At_i))$ returns the degree of match between the values of the two attributes.

4.6.1.5 The Central Control

The role of the Central Control is to request services from the neural network and the CBR system and to mediate their answers.

The main steps followed by the Central Control algorithm are described below:

1) Verify the number of previous transactions existing in the *case library* for the same account to which the incoming transaction belongs.

2) If the number of previous transactions < minimum number of cases[3].
   - Activate the neural network for the incoming transaction;
   - Give the answer provided by the neural network as a final answer;
   - Store the transaction in the *case library* with the appropriate classification result.

3) If the number of previous transactions >= minimum number of cases
   - Activate both the CBR system and the neural network for the coming transaction;
   - If the answers coming from the CBR system and the NN are the same, give them as a final answer;
   - If the answers are different, the final answer should be the one carrying the higher confidence factor:

$$Final = Max\,(\alpha \times CBR_{confidence\text{-}factor}\,,\ NN_{confidence\text{-}factor})$$

where $CBR_{confidence\text{-}factor}$ represents the highest degree of similarity computed in the CBR matching process, α represents a constant used to normalise the value of the result provided by the CBR system in relation to that provided by the neural network, and $NN_{confidence\text{-}factor}$ represents the output of the neural network.

4.6.1.6 Evaluation

The prototype of the system to control credit card transactions has been implemented in Common Lisp. A total number of 54 findings was described in the knowledge base for the 2 possible diagnoses (i.e. *to be granted* and *to be denied authorization*). A major British bank provided a database with 172 cases of credit card theft, each case containing an average of 18 transactions, totalising 3237 transactions. Approximately half of the total number of cases was used to train the neural network, and the other half was used to test the system. The system was able to classify correctly 89.0% of the cases (ie. 1430 transactions out of the 1606

[3]The parameter minimum number of cases has been set to several different values, the best performance being achieved when the parameter was set to numbers in the interval [5,9].

transactions contained in the testing set). The general performance of the system was considered to be satisfactory, especially because human specialists (who are not highly trained) are expected to produce a 90% level of performance. Furthermore, for our application, by having the two reasoning components working side by side, we could simulate the behaviour of humans performing the same classification task. When the first transactions for a particular case of credit card theft appear, the human expert can only exploit general knowledge about use and misuse of credit cards to classify this first set of transactions. However, after a certain number of transactions, the observer can start using more specific knowledge related to that particular case; that is to say, the instances of transactions carried out with the same card. The prototype described here behaves similarly, leaving the first set of transactions to be classified by the neural networks and only later activating the CBR system. Moreover, the use of a CBR component enables the system to retrieve previous transactions that can help the user to understand why a new transaction was granted or denied authorisation.

4.6.2 Comparing the experimental systems

Some similarities can be found between the system for the classification of credit card transactions (*Ccard*) and HYCONES. For example, the representation of the domain knowledge in a hierarchy of frames, and the use of *diagnosis descriptors* to represent in a symbolic formalism the knowledge used to build explanations for the reasoning process. The main differences between the two systems lie in the methods used to build the *diagnosis descriptors* and in the reasoning process itself. Table 4.2 shows the main differences among HYCONES, *Ccard*, NN-CBR and two other experimental systems.

	HYCONES (Leao & Reategui 1993)	*Ccard* (Reategui & Campbell 1995)	*ChartD2* (Reategui, Campbell & Leao 1996a)	*Profile* (Reategui, Campbell & Borghetti 1995)	NN-CBR (Reategui, Campbell & Leao 1996b)
Building *diagnosis descriptors*	Calculating the sensitivity and specificity of findings	Calculating the sensitivity and specificity of findings	Calculating the sensitivity and specificity of findings	Through the CNM network	Through the CNM network
Reasoning	CNM network	CNM network and CBR module working independently, the two being coordinated by a central control	*Diagnosis descriptors* used to make hypotheses and guide the search for similar previous cases	*Diagnosis descriptors* used to make hypotheses and guide the search for similar previous cases	Neural network used to make hypotheses and to provide remindings for the search of similar previous cases

Table 4.2: Comparing the experiments that led to the NN-CBR architecture

HYCONES built the *diagnosis descriptors* according to the computation of the specificity and sensitivity degrees of all the findings observed for each diagnosis. Its reasoning process was based on the recognition of patterns using only the generalised knowledge of the CNM network. The explanation of reasoning in HYCONES was based solely on the description of the importance of each finding in the diagnostic process (knowledge that could be obtained from the *diagnosis descriptors*). The idea of using cases as a complementary resource for the explanation of reasoning (Georgin et al. 1994) seemed to be appropriate. Furthermore, it could also improve the level of acceptability of answers provided by the system.

Based on this idea, the system for the classification of credit card transactions (*Ccard*) combined the main structure of HYCONES (ie. the frames hierarchy and the CNM) with a CBR component. *Ccard* made use of a central control unit to mediate the answers given by the neural network and the case-based component. The main problem with the architecture of *Ccard* was that its structure was somewhat connected to the application for which it was developed. Moreover, the CBR module used in *Ccard* was based on a nearest-neighbour algorithm with no indexing scheme, which we thought should be improved if we were to try a similar approach in other classification problems with large case libraries.

A different architecture was then conceived, following the same idea of investigating how the use of specific cases in CBR could be complemented by the help from other sources of knowledge. This architecture was implemented in the system *ChartD2*, which addressed the problem of congenital heart disease diagnosis. *ChartD2* proposed some solutions for the indexing problems faced in *Ccard* through the use of the *diagnosis descriptors* learned through the calculation of the sensitivity and specificity of findings. No neural network was used, though, and the *diagnosis descriptors* were the main components responsible for making hypotheses and for guiding the search in the library for similar cases. One of the main limitations found in this experiment was that a more powerful generalisation mechanism was needed to build the descriptors and to index the cases in the library.

Following *ChartD2*, an experiment in clinical psychology was built, where the main problem to be solved was the identification of the psychological *Profile* of patients in evaluation for heart transplants. In this experiment the same type of reasoning process conceived in *ChartD2* was kept. However, in *Profile* the *diagnosis descriptors* were built with the help of the CNM. Some further tests, however, showed that the CNM itself could be more useful than the *diagnosis descriptors* to index cases in the case library. The NN-CBR architecture has been the outcome of these last experiments. In this hybrid architecture, the CNM is used not only to build the *diagnosis descriptors* but also to make hypotheses and guide the CBR search for similar cases in the library. The next chapter details the NN-CBR architecture and each of its main components. Chapter 6 then presents validation results obtained from a series of tests with NN-CBR applied to the CHD domain as well as in three other domains.

Chapter 5

The Hybrid NN-CBR Model

This chapter introduces NN-CBR, a hybrid neural network and CBR model that combines specific and general knowledge in reasoning. Specific knowledge is represented in the form of cases while general knowledge is represented in neural networks as well as in diagnosis descriptors. When solving a new case, the NN-CBR system uses its general knowledge to guide the search for the most similar case it has already 'seen'. The retrieved case, representing specific knowledge, is then used to suggest a possible solution for the new case. NN-CBR (Reategui, Campbell & Leao 1996b) is the final result of a series of experiments on hybrid CBR systems (Reategui & Campbell 1995; Reategui, Campbell & Borghetti 1995; Reategui, Campbell & Leao 1996a). Besides enhancing some of its predecessor's capabilities, the structure and behaviour of NN-CBR offer solutions for common problems in CBR in general, such as case matching, indexing and learning.

5.1 Introduction

The premise that human thought is based on the use of past experience has been supported by research in both psychology and cognitive science. According to this premise, learning is a product of experiencing new facts, trying to understand them and, while doing so, integrating them with the knowledge we already have. New experiences can be integrated with our previous body of knowledge in two different ways. They can be compiled into summary representations or they can be incorporated in our memories as single episodes. Experiences that lie in the memory as single episodes will form a body of specific knowledge, where all the details about each experience are kept. The summary representations of experiences, on the other hand, form a body of more generalised knowledge that covers the normal but not situations that are different from the norm.

A memory system that attempts to model cognitive processes of the human brain should therefore be able to store and manipulate specific knowledge, in the form of single episodes, as well as more general knowledge, in the form of abstractions or summary representations of these episodes.

This chapter introduces one particular case-based model where both general and specific knowledge are used in reasoning. Specific knowledge is represented in the form of cases while general knowledge is represented in the form of *diagnosis descriptors* and neural networks. This model was used in the development system for the diagnosis of Congenital Heart Diseases (CHD). When finding a solution for a new case, the system uses its general knowledge to guide the search for the most similar case it has already 'seen', and to increase the reliability of the solution found. The retrieved case, which represents specific knowledge, is then used to indicate a possible solution and to build the final explanation.

The next section introduces NN-CBR, describing how each of the elements of the hybrid architecture operate.

5.2 The NN-CBR Model

The NN-CBR model has been built with the main purpose of investigating how CBR and neural networks can cooperate in order to solve problems observed in each approach. While in other previous research cases are usually kept as an integral part of the neural network, in our model the neural network and case-based components are independent. This feature enabled us to deal separately with problems related to CBR (e.g. indexing and retrieval) and neural networks (e.g. the interpretation of the mathematical knowledge of the network, a problem which has not been treated in previous published research on the combination of CBR and neural networks).

Figure 5.1 shows the main components of the hybrid architecture and the way they all interact.

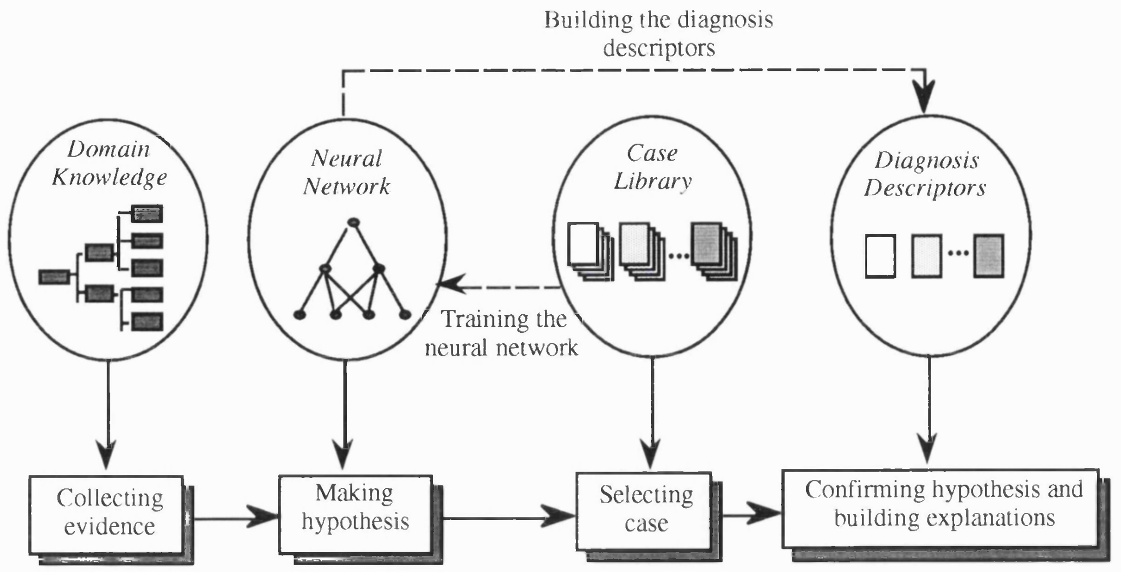


Figure 5.1: The main components of the hybrid NN-CBR architecture and their interaction

Four different components can be distinguished, each playing a different role in the reasoning process. The *domain knowledge* unit represents knowledge of the application domain by storing a hierarchical representation of all the findings[1] used in case descriptions. This unit is employed mainly for guiding the collection of information from the user and for inferring further evidence. The *case library* stores case descriptions and the solutions adopted in these cases. The *neural network* is trained with the cases stored in the library, and is used during the consultation process to make hypotheses of possible diagnostic solutions and to guide the search for similar cases. The *diagnosis descriptors* keep a summary representation of the knowledge stored in the neural networks, and are used mainly to ratify or refute a final result, and to build explanations.

The next subsections examine each of the components of the architecture.

[1]A finding corresponds to a feature/value pair. We have opted for using the word 'finding' with the purpose of employing more accurate terminology for the medical example presented later in this chapter.

5.3 Representing the Domain Knowledge

The domain knowledge unit serves the purpose of representing knowledge of the application domain. It keeps taxonomic knowledge about the findings used in case descriptions as well as storing interrelationships among findings. Here, the domain knowledge is represented in the same way as for HYCONES (Leao & Reategui 1993) and *ChartD2* (Reategui, Campbell & Leao 1996). The four classical abstraction concepts have been used to provide the necessary flexibility for knowledge modelling (Hull & King 1987).

Through the use of the abstraction concepts, namely generalisation, classification, and aggregation, the domain knowledge can be represented in a complex hierarchy where:

- an *instance* represents a basic finding that is used in case descriptions;
- a *class* describes the common aspects observed in all of its instances (classification);
- a *superclass* describes common aspects of all of its subclasses (generalisation);
- *aggregations* groups different classes that, together, represent a different concept. For example the parts of a car, when put together, represent the car itself.

Figure 5.2 presents an excerpt of the hierarchy used to represent knowledge in the domain of CHD. This hierarchy of findings has been constructed previously by Leao, Timmers, Van der Lei & Mulligan (1990). The class *Domain*, located at the top of the hierarchy, is divided into three subclasses, namely *History*, *Investigation examination* and *Physical examination.* Each of these classes is subsequently subdivided by *is-a* and *part-of* arcs. In the example, *is-a* arcs partition the class *Investigation examinations* into *Invasive* and *Non-invasive*, implying that the properties defined for *Investigation examinations* are also valid for its subclasses. The class *Non-invasive* is divided into two other classes to represent results obtained in *ECG* and *CXR* examinations. The class *ECG* is connected to the classes *Rhythm*, *Conduction disturbances* and *Hypertrophy of heart chambers* by *part-of* arcs, implying that an *ECG* examination should be made out of these three types of analysis. The class *Conduction disturbances* is not subdivided any further; instead, it is connected to an instance called *Left anterior hemi blockage.* Each of the instances of the domain hierarchy represents a medical finding that is symptomatic of one or more cardiac diseases. In the example, the instances *Down's syndrome*, *Cardiomegaly on CXR*, *Left anterior hemi blockage* and *Increased pulmonary flow on CXR* are defined as being symptomatic of *Atrioventricular septal defect* (AVSD), a typical problem in the domain of CHD.

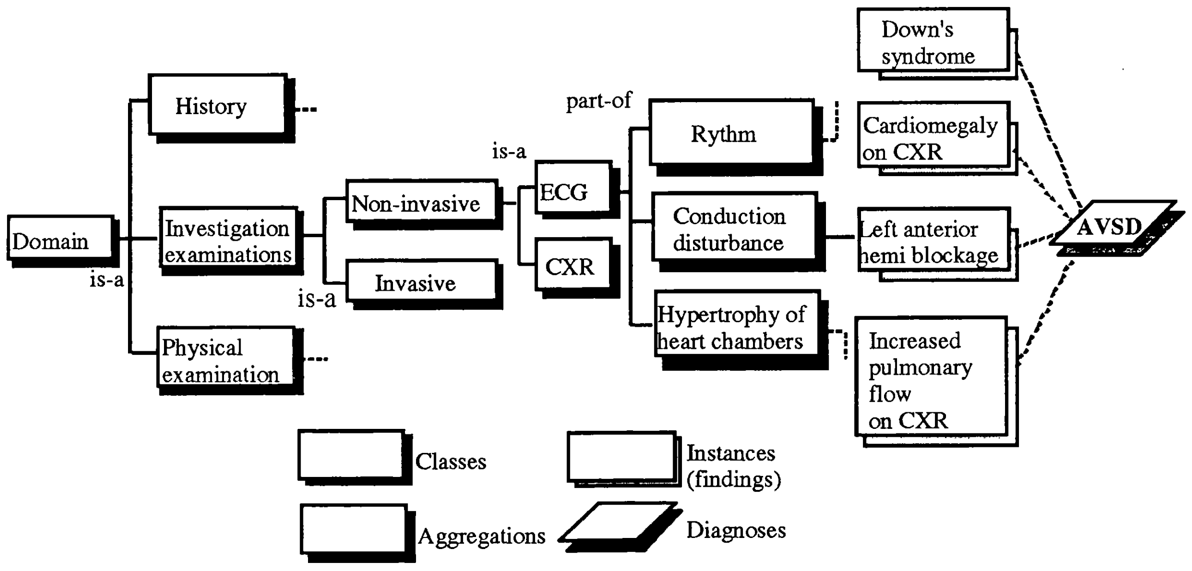


Figure 5.2: The representation of the domain knowledge

The instance described for each finding outlines its most important properties. For example, the finding *Left anterior hemi blockage* is described in the following way:

- it is-a type of *Conduction disturbance*;
- its morbidity is 4 on a 1-5 scale, which means that this finding is of considerable importance and would hardly ever indicate a false-positive result[2];
- it is part-of the results obtained from an *ECG* examination.

The description of findings can be used for different purposes, such as building explanations, inferring facts during the reasoning process, or simply for consultation. The inference of other facts during the reasoning process can be helpful in providing further evidence for incoming cases. For example, determining that the presence of the finding *Bi-*

[2]This is the morbidity scale used by the system Internist-I (Miller et al. 1986).

ventricular hypertrophy implies the presence of both *Right ventricle hypertrophy* and *Left ventricle hypertrophy* enables the system to conclude that the observation of the first finding indicates that the presence of the other two findings should be also taken into account.

5.4 Representing Cases

The idea behind CBR is to use concrete cases in the reasoning process. When presented with a new case, a CBR system attempts to retrieve in its case library the case (or cases) that most resemble the current case description.

In case-based systems, the case library is usually organised in one of the following ways: flat memory, shared-feature networks, discrimination networks and redundant discrimination networks[3]. The last three approaches cluster together similar cases, which makes retrieval more efficient. However, these techniques are disadvantageous from another point of view. They demand extra space to store the organisational information for the networks and they require more complex procedures for the inclusion of cases in the case library. In addition, they present problems in dealing with missing information and in keeping the network optimal when cases are added (Kolodner 1983).

Here, the case library is organised in a flat memory partitioned by the cases' diagnoses; that is, its cases are grouped according to their diagnoses and stored in a sequential manner. The main advantage of this approach is the simplicity of the case library, which facilitates its maintenance and allows for the use of simpler retrieval and learning algorithms. The major disadvantage of this approach is that, for big case libraries, the search for a case can be very expensive. In our NN-CBR approach, however, an indexing mechanism based on the use of remindings provided by the neural network reduces the search in the case library, thus minimising the problem of having the cases stored in a flat-memory fashion.

Each case consists of a simple feature/value vector (or list of findings) and a diagnosis. However, as a description of the domain is stored in the domain knowledge hierarchy, the representation of a case can be seen as structured. Figure 5.3 shows the representation for the case AVSD17, composed of the medical findings: *Left anterior hemi blockage*, *Wide and split*

[3]See (Kolodner 1993) for a detailed explanation and examples of actual use of each of these memory organisation schemes.

S2, *Increased pulmonary flow on CXR*, *History of cardiac murmur*, *Cardiomegaly on CXR*, *Systolic murmur*, *Frequent respiratory infections* and *Growth retardation.* These findings can be observed in the rightmost nodes of the hierarchy, ie. the *instance* nodes. The connections between findings and classes, and among the classes themselves, further explain the types of problems represented in the hierarchy. For example, from figure 5.3 we can work out that *Left anterior hemi-blockage* is a type of *Conduction disturbance*, which is *part-of* the results obtained from an *ECG*, which is a *non-invasive investigation examination.* The structured representation of cases can give the user a more thorough view of the problem, and make explicit the interrelationships among classes for the case findings.

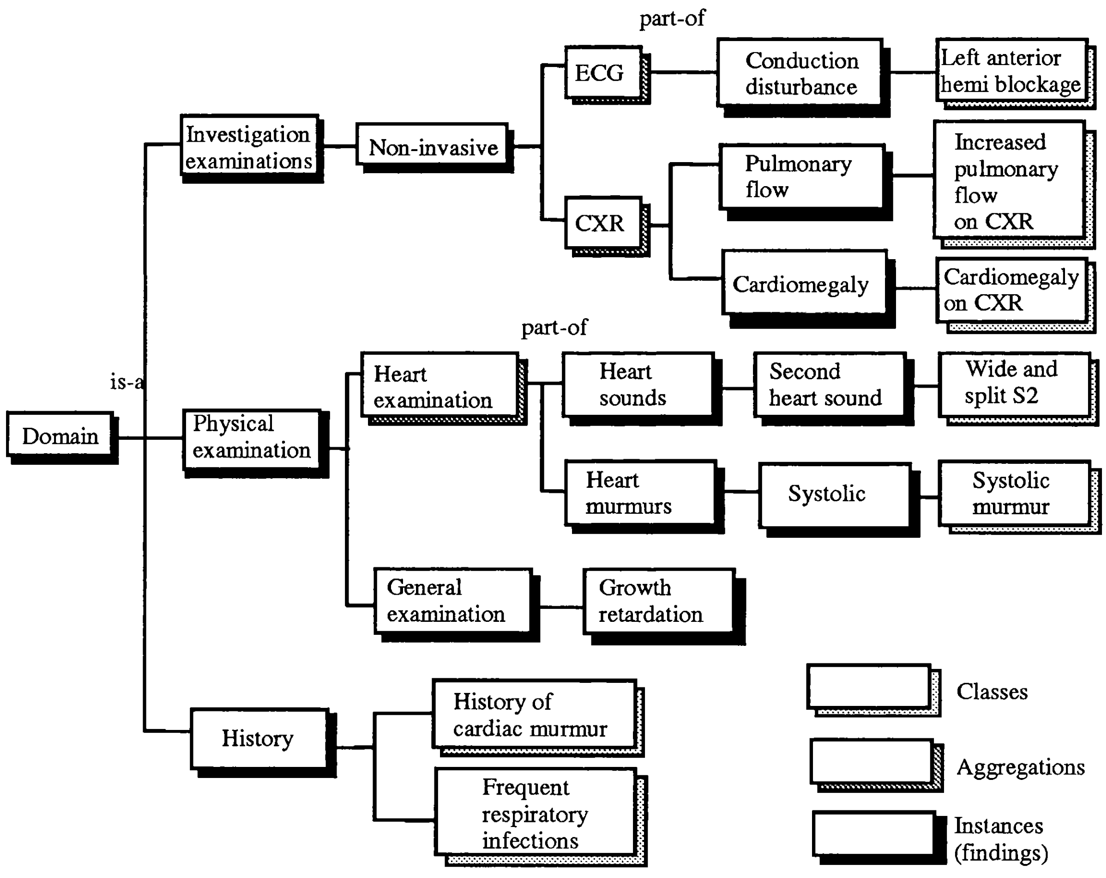


Figure 5.3: The structured representation of cases

5.5 The Combinatorial Neural Model

The CNM has been based on the knowledge representation formalism of knowledge graphs. Here, it is responsible for learning which findings and clusters of findings are important for each diagnosis. It has a feedforward topology with three layers[4], where:

- output layer: contains different nodes, each representing a separate diagnostic hypothesis;
- input layer: contains nodes that represent evidence such as symptoms, test results or any other information that supports the diagnostic hypothesis.
- intermediate layer: specifies different combinations of evidence that can lead to a particular diagnostic hypothesis.

Figure 5.4 depicts the basic structure of the CNM.

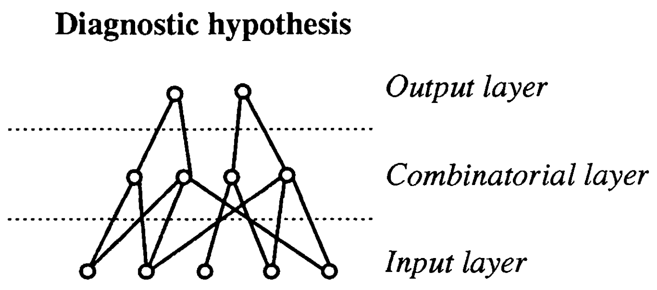


Figure 5.4: Basic structure of the CNM

The input layer is formed by fuzzy-number cells. These fuzzy numbers (values in the interval [0,1]) represent the degree of confidence the user has in the information that is observed and inserted into the neural network. Cells in different layers are linked by connections with an associated weight which represents the influence of lower-layer cells on the output of upper-layer cells.

[4]Although the CNM can assume a configuration of three layers or more, usually three layers are sufficient to represent the problems addressed.

The connections of the input layer can be either excitatory or inhibitory. An excitatory connection propagates the arriving signal using its weight as an attenuating factor. An inhibitory connection performs fuzzy negation on the arriving signal X, transforming it to 1-X. The connection then propagates the signal, multiplying the value obtained (1-X) by the connection weight.

The combinatorial layers are formed by hidden fuzzy AND-cells. They associate different input cells in intermediate chunks of knowledge which are relevant in the classification process. The output *Y* of the fuzzy-AND function corresponds to the minimal value coming from the lower layer, ie. the smallest value obtained from the product of individual input signals $x_1, x_2, \ldots, x_n$ with their corresponding connection weights $w_1, w_2, \ldots, w_n$ (figure 5.5).

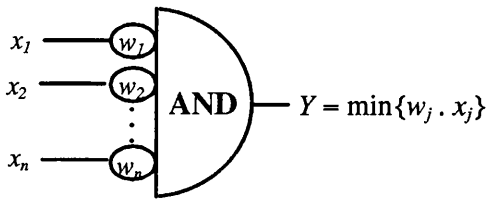


Figure 5.5: The fuzzy-AND function

The output layer of the CNM is formed by fuzzy OR-cells. They implement a competitive mechanism between the different pathways that reach the diagnostic hypothesis. The output *Y* of the fuzzy-OR function corresponds to the maximum value coming from the lower layer, ie. the largest value obtained from the product of individual input signals $x_1, x_2, \ldots, x_n$ with their corresponding connection weights $w_1, w_2, \ldots, w_n$ (figure 5.6).

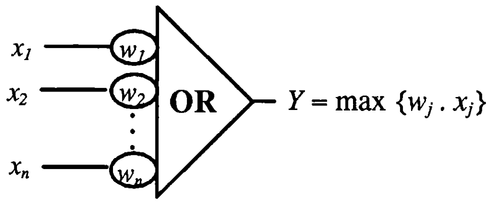


Figure 5.6: The fuzzy-OR function

5.5.1 The Neural Network Learning Mechanism

The learning mechanism of the CNM uses a supervised learning method to determine the combinations of features that are influential for each diagnosis. The learning method is split into two different algorithms:

- Punishment and reward algorithm
- Pruning and normalisation algorithm

The punishment and reward algorithm uses a mechanism analogous to that of backpropagation to identify successful and unsuccessful pathways of the neural network, ie. pathways that lead to the correct and to incorrect diagnoses, respectively. For each connection of the neural network that join a pair of nodes, two accumulators are defined: one to compute rewards, the other to compute punishments. The learning algorithm takes into account the *evidential flow* and the *importance of the example* when determining the values of punishments and rewards, where:

- the *evidential flow* observed in a connection corresponds to the following product (for excitatory connections):

  destination-node activation x *connection weight*

  For inhibitory connections, the following product is used:

  (1 - destination-node activation) x *connection weight*

- the *importance degree of an example* is a numerical value assigned to each of the training cases. The influence of an example in the learning process is directly proportional to its *importance degree.*

Two different versions exist for the punishment and reward algorithm. The first one, called the *scratch version,* is used to initialise a neural network and compute its punishment and rewards according to a set of examples. After going through all the training examples, the algorithm produces a trained network which stores punishment and reward accumulators in the interval $[-n_e,+n_e]$, for n_e the number of training examples given to the network. To become operational, the network has to have its accumulators converted into connection weights, which is the job of the *pruning and normalisation algorithm.*

The algorithm for *punishment and reward (refinement version)* is used to adjust the knowledge that already exists in the neural network, according to what the incoming examples contain. This version of the algorithm increments the punishment and reward accumulators taking into account the values previously calculated. As in the *scratch version*, the punishment and reward values produced have to be processed by the *pruning and normalisation* algorithm in order to make the neural network operational.

The normalisation process converts the net values of the accumulators (ie. rewards minus punishments) into connection weights with values between 0 and 1. This process generates an operational network. However, a large number of connections may still be removed from the neural network without influencing its performance. This is done by a pruning mechanism, which removes from the neural network all the negative or weak connections (ie. those with net values of the accumulators containing a negative or a small number). In practice, the pathways selected for pruning are those with connection weights smaller than a pruning threshold (which should not exceed *sqrt(Tacc)*, where *Tacc* is the minimum value to be accepted for an output of the network).

This learning algorithm identifies and maintains all the pathognomonic pathways (ie. those with no punishments and a positive number of rewards), even when a small net value of the accumulators is computed. It achieves this by attributing weights with values bigger than *sqrt(Tacc)* to these connections, which guarantees the acceptance of the hypothesis indicated by the pathway. The choice of thresholds is a very important detail for the good performance of the CNM network. The thresholds can be selected empirically through the systematic testing of the network with different threshold settings. The configuration producing the best performance is the one to be selected.

A more detailed explanation of the punishment and reward algorithm, and the pruning and normalization algorithm (Machado, Rocha & Denis 1992), is presented in Appendix B.

5.5.2 Training the CNM for a Simple Diagnostic Problem

Here we present an example to ilustrate the use of the CNM network in learning how to solve a simple diagnostic problem (Denis & Machado 1991). In this problem, only positive evidence is considered, which is frequently observed in medical diagnosis. The database used to train the system has four cases of patients with either disease *d1* or *d2*. The patients present some of the following symptoms: *s1, s2, s3, s4*. The cases that form the database are shown in table 5.1.

Patient	**Disease**	**Symptoms**
John	*d1*	*s1, s2, s3*
Diana	*d1*	*s1, s2, s4*
Maria	*d2*	*s1, s3, s4*
Peter	*d2*	*s2, s3, s4*

Table 5.1: The cases used to train the CNM network

When building the network for this particular example, four distinct input nodes (each representing a particular symptom) are created and linked to different combinatorial nodes.

When building the network for this particular example, four distinct input nodes are created, each representing a particular symptom. The input nodes are then connected to the diagnostic hypotheses either through a direct link, or through a combinatorial node. Each combinatorial node associates two or three input nodes, representing the importance of different combinations of symptoms for each hypothesis. Figure 5.7 shows the CNM network built for the example. Notice that the network has been split into 2 different drawings, one for diagnosis *d1*, the other for diagnosis *d2*, with the only purpose of making its visualisation clearer. The input nodes of both network excerpts are exactly the same, while their combinatorial nodes, as well as their output nodes, are different.

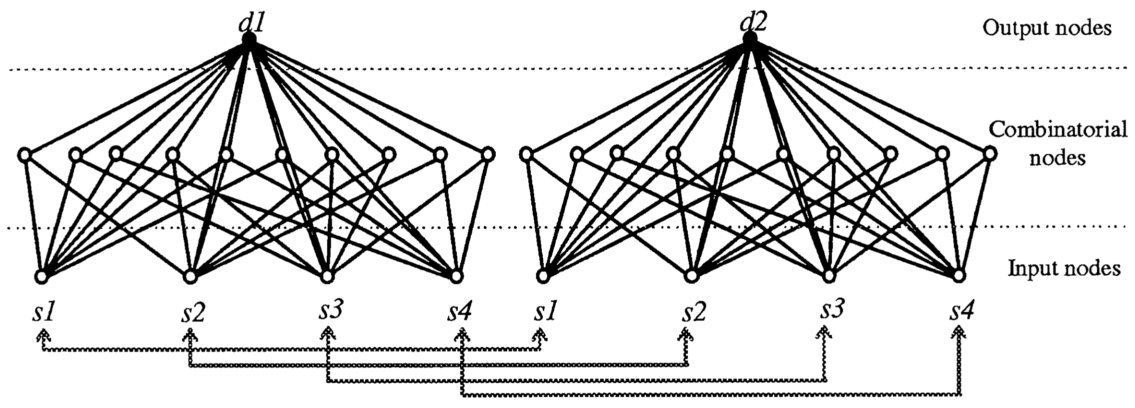


Figure 5.7: The initial neural network created for the example

The *scratch version* of the learning algorithm is then used to train the neural network. When presented with the first case of the database (that of *John*), the pathways activated by the evidence (*s1, s2, s3*) which lead to the correct diagnosis (*d1*) are rewarded by the scratch learning algorithm, whereas the pathways that lead to an incorrect diagnosis (*d2*) are punished. The top part of figure 5.8 shows the effect of the punishment and reward algorithm in the neural network when it is presented with the case of *John* (once again, with the purpose of simplifying the drawing, the neural network has been split into two different networks, one for diagnosis *d1*, the other for diagnosis *d2*).

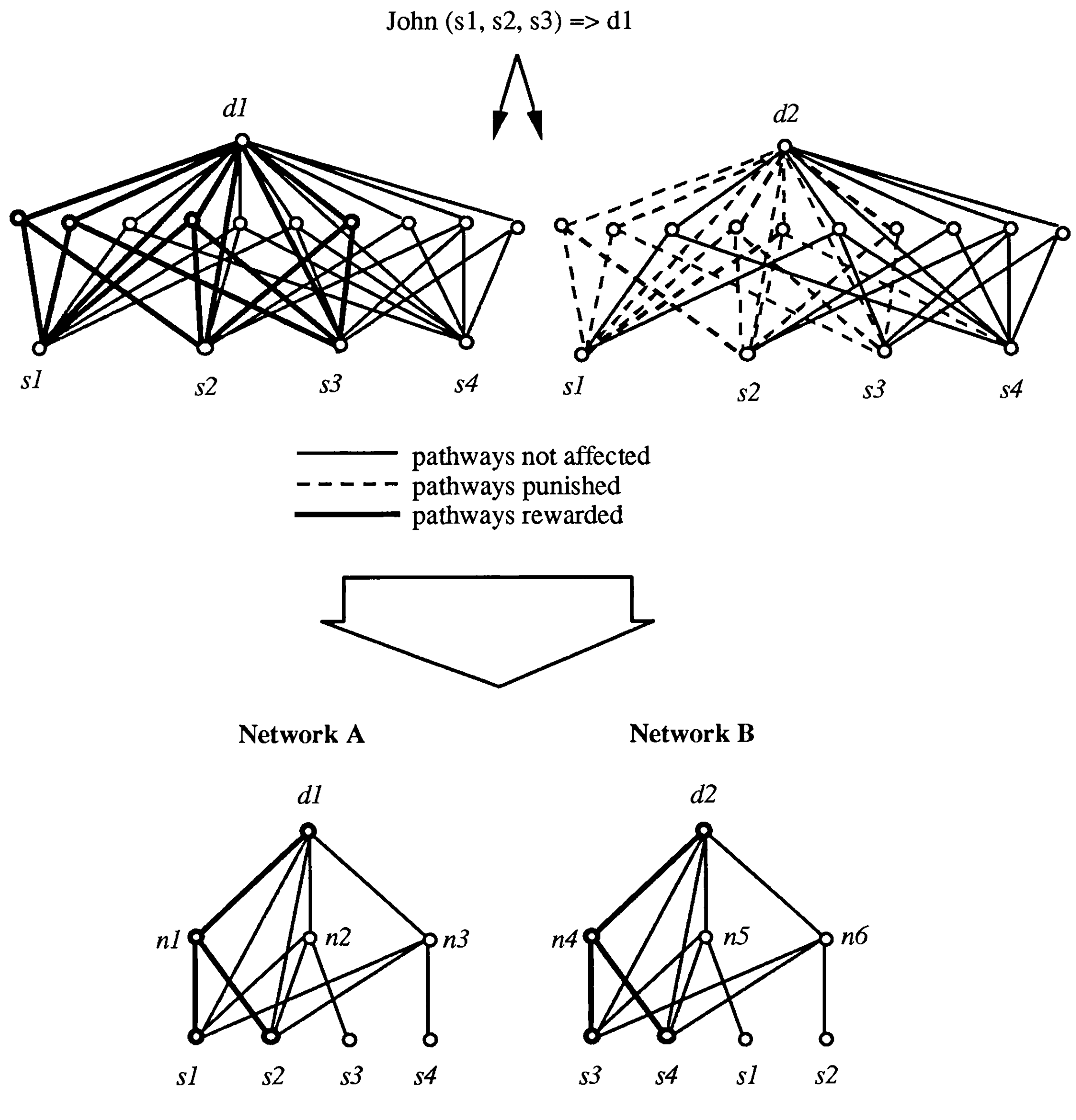


Figure 5.8: Training the neural network to solve a simple diagnostic problem

The bottom part of figure 5.8 shows the network already trained and pruned, where two different structures can be observed in the network. The node *n1* of *network A* represents a strong pathway leading to the diagnostic hypothesis, which has been called *knowledge germ.* The nodes *n2* and *n3* represent factual information about the patients John and Diana. However if a higher pruning threshold is used, only the strongest pathways will remain. For example, if the threshold 0.6 is used to prune the network, only the nodes *n1* of network *A*, and *n4* of network *B* would be kept.

5.6 The Diagnosis Descriptors

The main difficulty in representing the knowledge of the CNM neural network comes from fact that, for each diagnosis (or class) represented in the network, there is a large number of nodes in the combinatorial layer, each representing a combination of findings which is relevant for the identification of the diagnosis. Trying to describe the relevance of each node of the combinatorial layer in an 'all-or-nothing' symbolic fashion, would not result in a representation easy to read or understand, as the number of combinatorial nodes is usually large. For example, if we were trying to represent the knowledge stored in the non-pruned neural networks of figure 5.8, the following set of rules would be necessary.

- If *S1* Then *D1*
- If *S2* Then *D1*
- If *S1* and *S2* Then *D1*
- If *S1* and *S2* and *S3* Then *D1*
- If *S1* and *S2* and *S4* Then *D1*
- If *S3* Then *D2*
- If *S4* Then *D2*
- If *S3* and *S4* Then *D2*
- If *S1* and *S3* and *S4* Then *D2*
- If *S2* and *S3* and *S4* Then *D2*

We can conclude from this example that the rules created to explain the two neural networks are not much easier to read than the networks themselves. And for more complex problems, the number of rules would increase dramatically, making rule-sets even more intricate and difficult to understand. The solution conceived for representing the knowledge of the CNM has been to describe, for each problem addressed, the importance of the findings that are frequently observed. The symbolic structure created for representing the knowledge of the CNM has been called *diagnosis descriptor.* A *diagnosis descriptor* is a record-like structure which was based on the ideas taken from the mechanism for knowledge representation of Knowledge Graphs (KGs) (detailed in the previous chapter). Figure 5.9

shows the main structure of a *diagnosis descriptor*.

Diagnosis descriptor	***disease name***
Trigger:	*f1*
Primary:	*f2, ..., fm*
Secondary:	*fm+1, ..., fn*

Figure 5.9: The main structure of the diagnosis descriptor

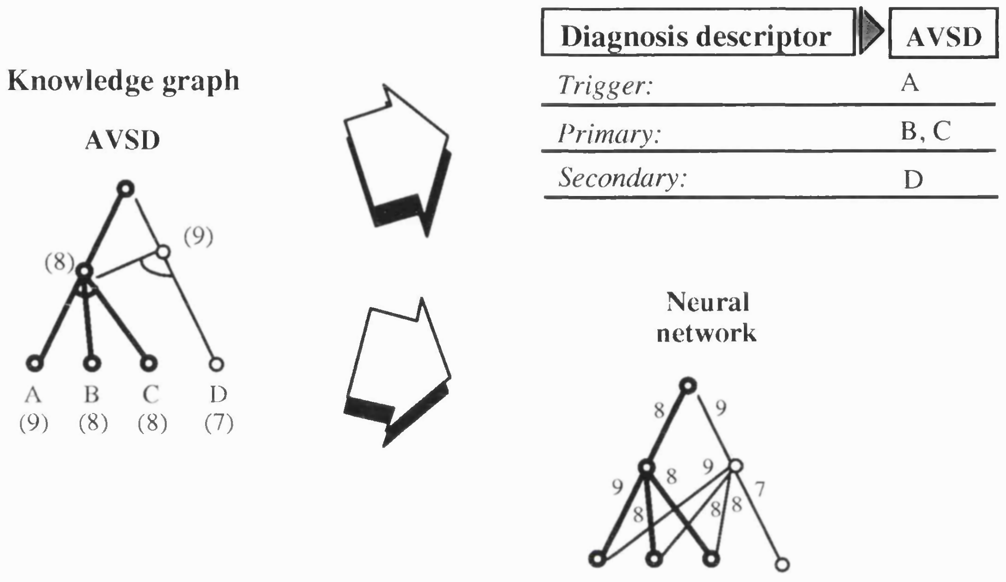
The descriptors rank the most important findings for each diagnosis in three attributes: *trigger*, *primary findings* and *secondary findings*. The findings referenced by these attributes are characterised by their frequency and specificity in relation to the diagnosis. Findings that are specific are important for their discriminatory properties, being normally observed only in one diagnosis and not in others. Findings that are frequent, on the other hand, are important for confirming a certain diagnostic hypothesis. The description of each attribute follows:

- **Trigger:** this attribute references a finding that is usually specific and frequent for the diagnosis and that, when present, strongly indicates the diagnosis as a potential solution for the case. The trigger can be seen as the leftmost node in a knowledge graph, representing the most important evidence for the diagnosis.

- **Primary findings:** this attribute references a number of findings which are frequent for the diagnosis, and which usually appear together with the *trigger*. These findings correspond to the leftmost tree of findings which is observed in the knowledge graphs. *Primary findings* are important elements in the diagnostic process, being able to provide a high degree of confidence for the diagnostic conclusion;

- **Secondary findings:** this attribute references findings that are not so frequent or specific for a diagnosis, but which can can reinforce the diagnostic conclusion (or sometimes determine it). These are other findings found in the knowledge graph, findings that do not belong to the graph's leftmost tree, but which can also be used as evidence for reinforcing a diagnosis.

A mapping between knowledge graphs, *diagnosis descriptors* and neural network has been defined here, as the *diagnosis descriptors* as well as the CNM have been based on the reasoning model of the knowledge graphs. Figure 5.10 gives an example of this mapping, where a knowledge graph for the diagnosis of AVSD is converted into a corresponding *diagnosis descriptor* and a neural network.



Diagnosis descriptor	**AVSD**
Trigger:	A
Primary:	B, C
Secondary:	D

Figure 5.10: The mapping of a KG into a diagnosis descriptor

The mapping from knowledge graph into neural network is direct. The arcs linking nodes of the graphs are transformed into neural-network connections, while the importance degrees assigned to the nodes of the graphs are converted into connection weights. The mapping between graphs and *diagnosis descriptors* is also straightforward. The attributes *trigger*, *primary findings* and *secondary findings* of the diagnosis descriptors correspond to the following elements of the knowledge graphs, respectively: leftmost node, leftmost tree[5] and other nodes.

[5]To simplify the representation of the *diagnosis descriptors*, the finding corresponding to the *trigger* of the diagnosis is not repeated in the list of *primary findings*.

The ideas for converting knowledge represented in one form (knowledge graphs) into another form (eg. neural network or *diagnosis descriptors*) have been used to provide the guidelines for the interpretation of the neural network knowledge into *diagnosis descriptors.* The main goal here has been to transform the mathematical knowledge stored in the neural network, in the form of accumulators and connection weights, into more intelligible symbolic knowledge. The following method is used to select the findings for each attribute:

- **Trigger**: references a finding that is normally observed in a diagnosis but that is not observed in others, being highly specific and frequent. The findings selected as triggers are those that appear in the nodes of the network possessing the highest connection weights. For instance, for the diagnosis AVSD, the node of the network containing only the finding *Left anterior hemi blockage* is the strongest node. Thus, the findings *Left anterior hemi blockage* is selected as the *trigger* for the diagnosis AVSD. When more than one finding appears in the strongest node for one diagnosis (ie. the node with the highest connection weight), the finding with the highest specificity degree is selected as the *trigger.*

- **Primary findings**: this attribute references findings that have a high sensitivity degree, ie. findings that are very frequent for the diagnosis. This is justified by the fact that the findings observed in most of the cases for a particular diagnosis are generally expected to be present in other cases with that same diagnosis. These findings appear in nodes of the neural network with a high value in the reward accumulator. Additionally, they may also be observed in other nodes combined with the *trigger* (the *trigger* being another frequent finding). For instance, *Increased pulmonary flow on CXR* is selected as a *primary finding* for the diagnosis *Atrial Septal Defect* (ASD), because the node combining the finding *Increased pulmonary flow on CXR with* the *trigger* of ASD (ie. *Wide and Split S2*) has a high value in the reward accumulator.

- **Secondary findings**: this attribute references findings that reinforce the diagnostic hypothesis. They are selected by taking the findings that appear in strong nodes of the network, whether they have been combined with the *trigger* or not.

By discriminating the different levels of influence that findings have in relation to diagnoses, it is possible to represent, in a symbolic way, the importance of feature/value pairs (findings) in the identification of a diagnosis (or class). This could not be done by other hybrid architectures which also addressed the problem of representing in a symbolic system the knowledge of the neutal networks (e.g. NEXTOOL, which represented any connection of the neural network in the same type of *influence links* in its symbolic semantic network).

In the NN-CBR architecture, the *diagnosis descriptors* are employed in three important tasks. Firstly, for consultation purposes. A final-user who wants to know which findings influence the diagnosis of certain diseases may browse the *diagnosis descriptors* to obtain such information. For the system developer, this feature is also most important, as he or she can have an idea of the knowledge stored in the neural network by looking at the *diagnosis descriptors*.

The second function of the *diagnosis descriptors* is that of building explanations. When a conclusion is reached, the attributes of the *diagnosis descriptors* can be used to show which of the features observed in the problem case had more influence in the identification of a diagnosis. In a case-based context, the attributes of the *diagnosis descriptors* can be used as corroborative evidence for the user in the selection of a certain case as the most similar previous experience. For instance, if the new case and the most similar case retrieved have a set of common findings that are defined as *trigger* or *primary findings* in the *diagnosis descriptor*, it is likely that the two cases represent a good match.

The last function of the *diagnosis descriptors* is that of reinforcing a final result. This is important here especially because in the hybrid NN-CBR model we disregard the minimum degree of confidence accepted by the CNM neural network (we consider hypotheses coming from the network with a high or with a low degree of confidence). The mechanism for reinforcing a final result computes the average intra-class similarity for all cases in the case library taking into account the attributes *trigger* and *primary findings* of the *diagnosis descriptors*. A diagnostic hypothesis is refuted if its similarity with the problem case is smaller than the average intra-class similarity previously computed for the same diagnosis. This method for reinforcing a final result by using the knowledge stored in the *diagnosis descriptors* is presented with more detail in section 5.9.

The system HYCONES also addressed the problem of representing the knowledge of the neural network in symbolic *diagnosis descriptors.* The system used the CNM network to lead the reasoning process, and built the *diagnosis descriptors* through the computation of the sensitivity and sensibility degrees of the findings observed in a set of training examples. Here, the method for building the *diagnosis descriptors* has tried to be more faithful to the knowledge of the neural networks. Instead of using the sensitivity and sensibility concepts, we have considered directly the knowledge stored in the CNM network.

5.7 The Learning Process

In the hybrid NN-CBR approach, learning encompasses three main tasks[6]:

(1) training the neural networks;

(2) incorporating new cases in the case library;

(3) building the *diagnosis descriptors.*

The neural network is initially trained according to the learning algorithms detailed in the section 5.5. The cases used to train the network are also stored in the library in a sequential manner, as described in section 5.4. Then, the connections of the neural network are analysed and interpreted in *diagnosis descriptors*, in order to provide the system with symbolic representations of the diagnostic knowledge stored in the network.

We present here an example of the training of the neural network in the domain of CHD, as well as the construction of *diagnosis descriptors* for the diagnoses considered. Figure 5.11 shows an excerpt of a neural network computing punishments and rewards for the diagnoses of AVSD and VSD when case AVSD7 is presented.

[6]One more step could be included here, which is the knowledge-acquisition phase necessary for the construction of the domain knowledge hierarchy. Learning this type of declarative knowledge involves a knowledge engineering process where experts have to be interviewed and textbooks consulted. In Hull & King (1986) and Mattos (1991) there are more detailed explanations about the acquisition and representation of declarative knowledge using the abstraction concepts.

The excerpts selected show the influence of four different findings in the identification of the two diagnoses.

Case presented: **AVSD7**

Left anterior hemi blockage
Down's syndrome
Increased pulmonary flow on CXR
Systolic cooing murmur right ventricle
History of cardiac murmur
Cardiomegaly on CXR
Left atrium hypertrophy on ECG

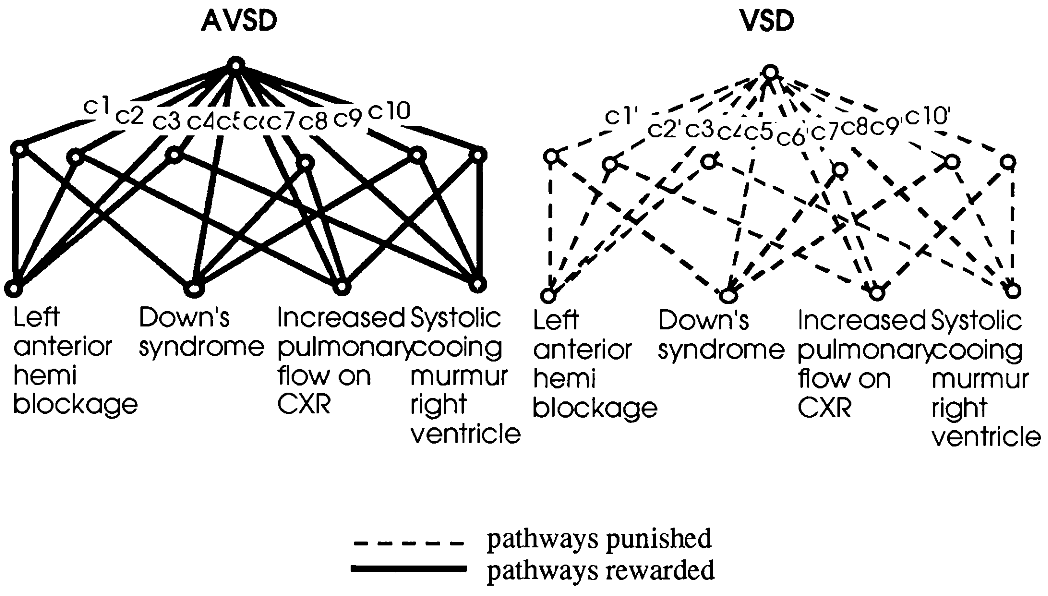

AVSD
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10
Left anterior hemi blockage
Down's syndrome
Increased pulmonary flow on CXR
Systolic cooing murmur right ventricle
VSD
c1' c2' c3' c4' c5' c6' c7' c8' c9' c10'
Left anterior hemi blockage
Down's syndrome
Increased pulmonary flow on CXR
Systolic cooing murmur right ventricle
pathways punished
pathways rewarded

Figure 5.11: The punishment and rewards computed by an AVSD and a VSD network when case AVSD7 is presented

In order to simplify the drawing, the network has been separated into two small networks, both with the same structure. The input and intermediate nodes in both networks represent the same groupings of findings. The difference between them is that the output node of the first network represents the AVSD diagnosis, while the output node of the second network represents the VSD diagnosis.

When presented with the case AVSD7, all the connections of the AVSD network are activated, leading to the correct diagnosis. The connections of the VSD network are also activated, but they lead to an incorrect diagnosis (VSD instead of AVSD). Because of this the connections *{c1, c2,..., c10}* are rewarded, while *{c1', c2',..., c10'}* are punished.

After the training period, the connections which have a larger number of punishments than rewards are removed from the network. In the given example, the connections *{c1, c2, c3, c5, c7}* still remain in the AVSD network, having reward values higher than punishment values. The connections *{c8', c10'}* have not been rewarded in observation of case AVSD7. However, the findings represented in these connections are very common in VSD cases, which causes a high number of rewards to be computed by the connections. Table 5.2 shows the final number of punishments and rewards for these remaining connections, after the complete training period (66 training examples).

	c1	*c2*	*c3*	*c5*	*c7*	*c8'*	*c10'*
rewards	8.55	7.22	10.26	10.26	7.22	15.4	13
punishments	0	0	0	1.71	1.44	0.85	0.72
weight	0.93	0.89	1	0.83	0.56	1	0.84

Table 5.2: The punishments and rewards for the AVSD and the VSD networks

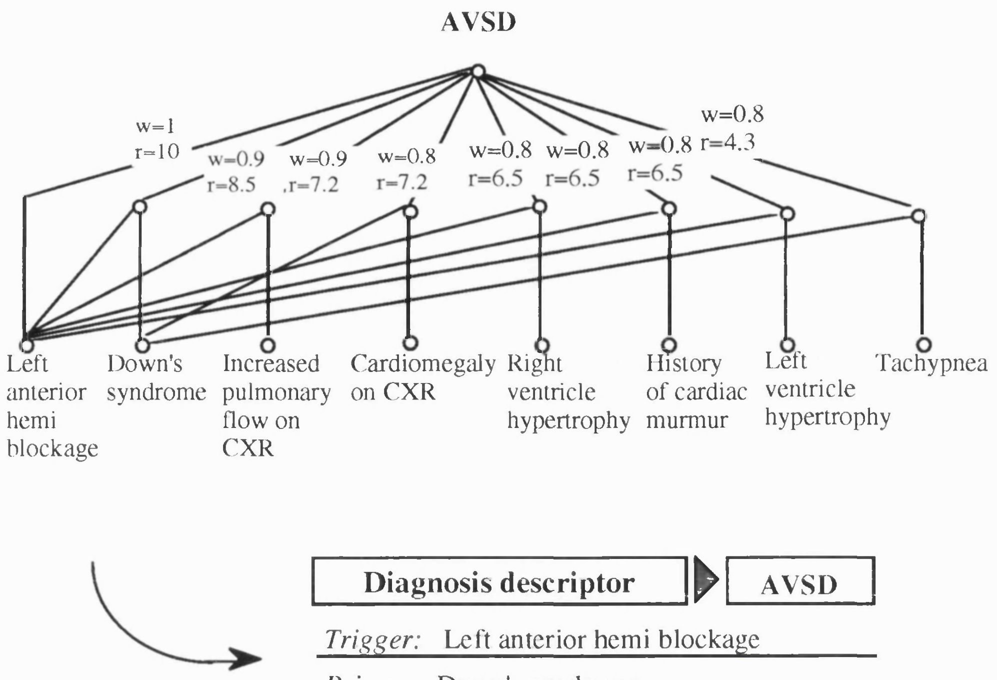
The punishment and reward accumulators of the network are then normalised into connection weights between the values 0 and 1. The following information can be observed or concluded from the table:

- The connection *c3* (from *Left anterior hemi blockage* to AVSD) obtained the highest weight for the AVSD diagnosis (1.0), which indicates that, when present, this finding has a very high importance for the identification of AVSD. For the VSD problem, the connection *c8'* obtained the highest weight.
- The nodes connecting the *Left anterior hemi blockage* to other findings (*Down's syndrome* and *Increased pulmonary flow on CXR)* also obtained high connection

weights (connections *c1* and *c2*).

- The finding *Increased pulmonary flow on CXR*, although frequent in AVSD or VSD problems, had its connections pruned both from the VSD and the AVSD network. This is justified by the fact that this finding is not specific to any of the diagnoses, and on its own it cannot indicate any of them. However, the finding *Increased pulmonary flow on CXR* appears in other nodes, where it is combined with *Left anterior hemi blockage*, as well as with *Systolic cooing murmur right ventricle* . This shows that the combination of *Increased pulmonary flow on CXR* with *Left anterior hemi blockage* is both frequent and speccific to AVSD, which is important for the identification of the diagnosis. When combined with *Systolic cooing murmur right ventricle, Increased pulmonary flow on CXR* is frequent and specific to VSD problems.

The construction of the *diagnosis descriptor* for the AVSD disease is shown in figure 5.12.


AVSD
w=1
r=10
w=0.9
r=8.5
w=0.9
r=7.2
w=0.8
r=7.2
w=0.8
r=6.5
w=0.8
r=6.5
w=0.8
r=6.5
w=0.8
r=4.3
Left anterior hemi blockage
Down's syndrome
Increased pulmonary flow on CXR
Cardiomegaly on CXR
Right ventricle hypertrophy
History of cardiac murmur
Left ventricle hypertrophy
Tachypnea

Diagnosis descriptor ▶ **AVSD**

Trigger: Left anterior hemi blockage

Primary: Down's syndrome
Increased pulmunary flow on CXR
Cardiomegaly on CXR

Secondary: Right ventricle hypertrophy
History of cardiac murmur
Left ventricle hypertrophy
Tachypnea

Figure 5.12: The construction of the diagnosis descriptor of AVSD

The finding with the highest connection weight *(w)* is selected as the *trigger* of the diagnosis (ie. *Left anterior hemi blockage*). Then, other findings that appear in nodes with a high reward accumulator *(r)* are selected as *primary findings*. In the example, the nodes combining the finding *Left anterior hemi blockage* with *Down's syndrome*, and with *Increased pulmonary flow on CXR*, are selected to form the *primary findings*. The node combining *Cardiomegaly on CXR with Down's syndrome* is also selected for its high frequency in AVSD cases *(r=7.2*, the same as *Left anterior hemi blockage* combined with *Increased pulmonary flow on CXR*). Other findings that appear in strong nodes are selected as *secondary findings*. For example, the finding *Tachypnea*, which appears in a node with *Down's syndrome*, is selected as a *secondary finding*.

It is important to remark here that, by eliciting the information above, we convert the 'opaque' knowledge stored in the network into meaningful attributes, which rank the importance of findings in determining the solutions of the problems addressed. This more intelligible representation can be used for several purposes, such as for consultation and for building explanations.

5.8 The Reasoning Process

Before starting the reasoning process, the user has to provide the system with all the evidence observed for the case to be solved. The collection of evidence from the user is guided by the knowledge stored in the domain knowledge hierarchy. The user enters *investigation examination* results, *physical examination* results, and *history* findings according to the terms used in the domain knowledge hierarchy.

The collected evidence is then presented to the CNM network. In the hybrid NN-CBR approach, three possibilities are considered:

1. The neural network is activated, according to the functions described in section 5.5, and indicates only one diagnosis as a possible answer, with a high or low degree of confidence[7]. In this circumstance, the CBR module has to find a case in the library that is similar to the new case and that can support the same diagnostic hypothesis. The CBR module uses the findings that have activated the strongest pathway in the network as remindings for the retrieval of cases from the library. A most similar case among these is found using a particular case-matching procedure. This procedure uses a similarity metric that is based on the computation of the *relative goodness* of attribute-value pairs (in our case findings). The *relative goodness* of an atttribute-value pair was used originally in INFERULE (Uthurusamy, Fayyad & Spangler 1991) as a measurement to support partitioning of a set of examples into two subsets. Instead of relying on the information

[7]When used on its own, the neural network does not present any result if the confidence degree that is obtained is lower than a preset threshold.

entropy to determine the discriminating power of an attribute (as in ID3[8] (Quinlan 1986)), the importance of an attribute-value pair in INFERULE is indicated by the fact that the class distribution in its corresponding subset differs significantly from the class distribution in the original set. In other words, the goal of this method is to choose an attribute that maximises the difference between the class distribution of the resulting subsets and the expected class distributions. In general, the bigger this difference, the more likely it is that the subset partition induced by the attribute-value pair is relevant to the classification. However, a small distance indicates that the distribution of classes in the original set does not change significantly when the set is partitioned, This would indicate that the attribute-value pair is probably not relevant to the classification task.
The *relative goodness* of an attribute-value pair can be obtained by the following method:

- Let C be a set of previous cases. Each case consists of a specification of the values of all attributes and a diagnosis, which can assume one of m possible values $\{d_1,..., d_m\}$;
- Let f_i be a finding equivalent to an attribute-value pair $<A=a_i>$ (where A is the attribute that takes on one of the values $\{a_1,..., a_k\}$);
- Let n_j be the number of examples in C that belong to diagnosis d_j. $C(f_i)$, contained in C, is the subset of examples in C with findings f_i. $C(f_i, d_j)$, contained in $C(f_i)$ is the subset of examples with finding f_i and diagnosis d_j.

[8]In ID3, the entropy measures the discriminating power of an attribute by favouring attributes that result in outcome vectors that are unevenly distributed. Thus, an attribute-value pair is not appropriate for partitioning a dataset if its corresponding subset of examples has equal numbers of examples from different classes. INFERULE's attribute selection criterion appeared to give the best overall results in comparison to several other methods that were tested, including ID3, the Chi-Square test, and Fisher's exact test for statistical independence (Uthurusamy, Fayyad & Spangler 1991).

Then, we have:

$$E_i(d_j) = \frac{|C(f_i)|}{|C|} n_j$$

where $|C|$ is the number of examples in the set C. $E_i(d_j)$ is the expected number of examples in $C(f_i)$ that have diagnosis d_j; so, $E_i(d_j)$ is an estimate of the actual value $|C(f_i, d_j)|$.

$$SE(f_i) = \sqrt{\sum_{j=1}^{m} \frac{E_i(d_j)(n_j - E_i(d_j))}{n_j}}$$

$SE(f_i)$ is a standard error mesurement associated with the E_is (estimates) defined previously. SE adjusts for the fact that an estimate that is based on a smaller data set is less accurate than one based on a larger set. The geometric distance between two class vectors is defined as:

$$DI(f_i) = \sqrt{\sum_{j=1}^{m} [\, E_i(d_j) - |C(f_i, d_j)| \,]^2}$$

$$<E_i(d_1), E_i(d_2), \ldots, E_i(d_m)> \text{ and } <|C(f_i, d_1)|, |C(f_i, d_2)|, \ldots, |C(f_i, d_m)|>$$

Note that the first vector is the expected class vector of the subset $C(f_i)$, and the latter is the actual class vector of $C(f_i)$. Finally,

$$Rg(f_i) = 1 - \left(SE(f_i) \,.\, \frac{1}{DI(f_i)} \right)$$

where $Rg(f_i)$ is the *relative goodness* of the findings f_i (or the attribute-value pair $<A=a_i>$). After evaluating all attributes and their values, the finding with the maximum Rg value is the one considered to be better for partitioning the data set. In our case, the findings with the biggest values of Rg are the ones considered to be more important for case comparisons.

The degree of similarity between two cases is then calculated by the weighted sum of the *relative goodness* of each finding that appears in both cases. The following formula is used:

$$\frac{\sum_{i=1}^{n} Rg(f_i)}{\sqrt{N_{findings}}}$$

where $Rg(f_i)$ is the *relative goodness* of a finding (i ranging through the n common findings between the new case and the previous case), and $N_{findings}$ is the mean of the number of findings of both cases. The use of $N_{findings}$ in the divisor expression of the formula accounts for unmatched features between two cases (the smaller the number of unmatched features, the highest the similarity). Moreover, using the square-root of $N_{findings}$ (instead of uniquely $N_{findings}$) enables cases showing a larger number of common features to have a higher similarity.

After finding the previous case that shows the highest similarity with the problem case, the diagnostic conclusion has to be confirmed. We do this by computing the similarity of its *diagnosis descriptor* with the problem case. A minimum similarity degree has to be found between the descriptor and the new case for the diagnosis to be given as a final result. The calculations used to reinforce a diagnostic conclusion using the *diagnosis descriptors* are presented in the next section (5.9).
An example of how the system would solve a case according to this first prescription is given below.

Case Presented - ASD32 : *Wide and split S2*

Increased pulmonary flow on CXR

Systolic ejection murmur pulmonary artery

Right ventricle hypertrophy on ECG

History of cardiac murmur

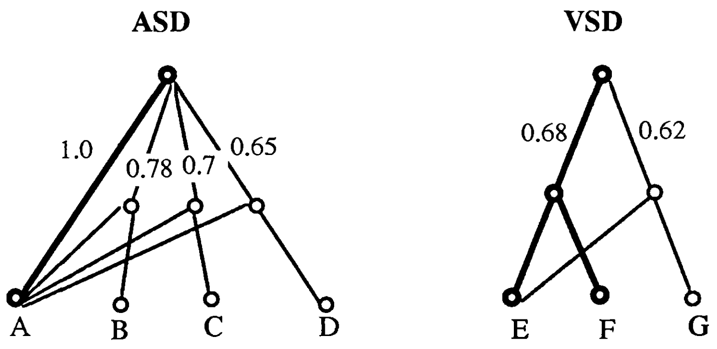
Cardiomegaly on CXR

Right atrium hypertrophy on ECG

Growth retardation

Extertional Dyspnea

When presented with case ASD32, the first step in the reasoning process is to activate the neural network. Figure 5.13 shows the strongest nodes of the network having been given case ASD32.



Strongest network nodes reaching the ASD hypothesis

A = Wide and split S2
B = Increased pulmonary flow on CXR
C = Systolic ejection murmur pumonary artery
D = Right ventricle hypertrophy on ECG

Strongest network nodes reaching the VSD hypothesis

E = Growth retardation
F = Exertional dyspnea
G = Right atrium hypertrophy on ECG

Figure 5.13: The strongest outputs of the neural network when presented with case ASD32

Two diagnoses are indicated, ASD and AVSD. However, the connection linking the finding *Wide and split S2* to the diagnosis ASD is the one to carry the highest weight (1.0), which produces the strongest output. The network is thus able to indicate the diagnosis of ASD with a high degree of confidence. The CBR module then has to retrieve cases in the library to support this hypothesis, ie. cases for the diagnosis of ASD containing the finding *Wide and split S2*. Figure 5.14 shows a few similar cases retrieved.

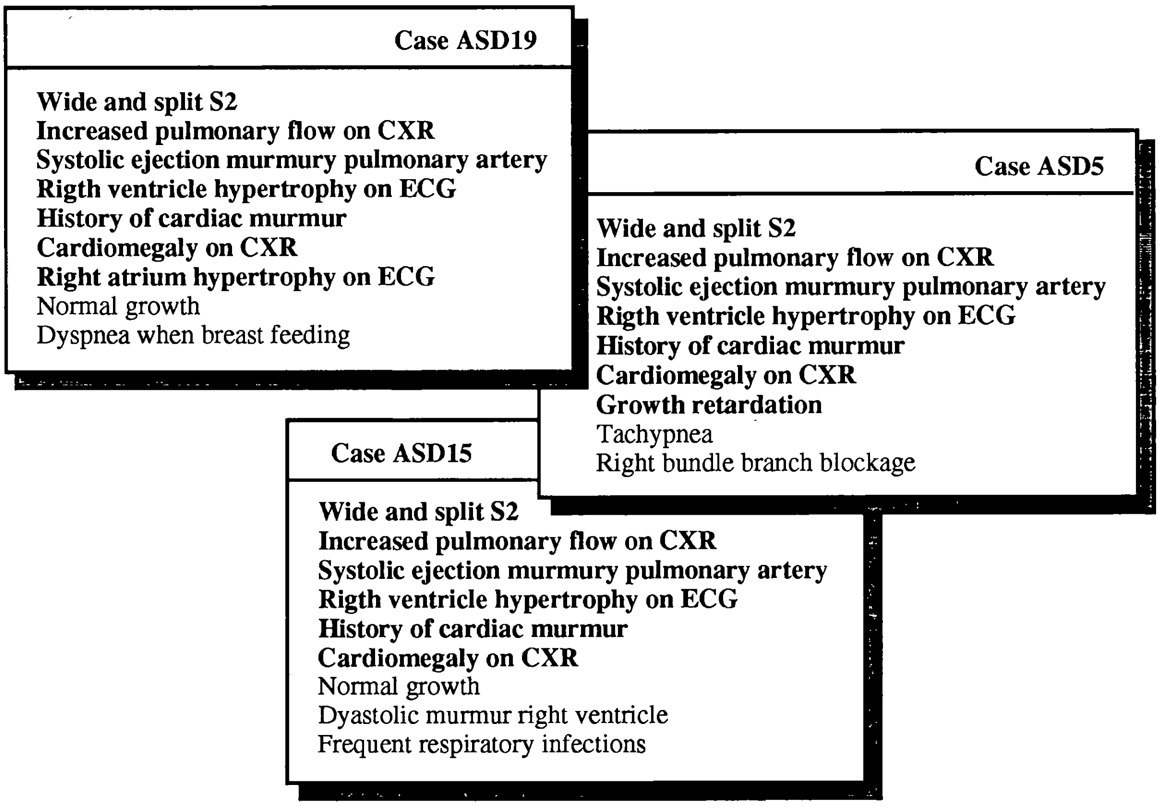

Case ASD19
Wide and split S2
Increased pulmonary flow on CXR
Systolic ejection murmury pulmonary artery
Rigth ventricle hypertrophy on ECG
History of cardiac murmur
Cardiomegaly on CXR
Right atrium hypertrophy on ECG
Normal growth
Dyspnea when breast feeding
Case ASD5
Wide and split S2
Increased pulmonary flow on CXR
Systolic ejection murmury pulmonary artery
Rigth ventricle hypertrophy on ECG
History of cardiac murmur
Cardiomegaly on CXR
Growth retardation
Tachypnea
Right bundle branch blockage
Case ASD15
Wide and split S2
Increased pulmonary flow on CXR
Systolic ejection murmury pulmonary artery
Rigth ventricle hypertrophy on ECG
History of cardiac murmur
Cardiomegaly on CXR
Normal growth
Dyastolic murmur right ventricle
Frequent respiratory infections

Figure 5.14: A set of cases similar to ASD32 retrieved by the case-matching procedure

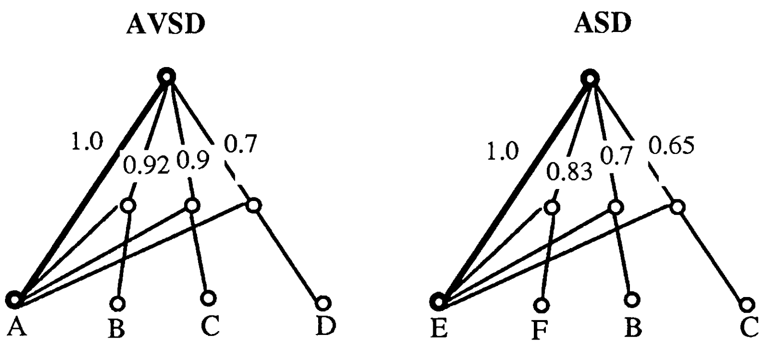
Cases ASD19 and ASD5 have the same number of similar and mismatching findings (matching findings appear in the figure in bold). However, some of the similar findings have a different degree of importance. This is taken into account by the matching procedure detailed previously, which calculates the importance of each finding through the computation of their *relative goodness*. The finding *Growth retardation* (0.67) has a lower *relative goodness* value than the finding *Right atrium hypertrophy on ECG* (0.73). Therefore, the case containing this last finding is selected as the most similar previous case (in the example, the case ASD19). In the last step of the reasoning process, the problem case is compared with the *diagnosis descriptor* of ASD, in order to reinforce the final diagnosis.

2. In the second path that may be followed by the reasoning process, the neural network finds two or more possible diagnoses with the same level of confidence. Normally, these hypotheses would be discarded by the neural network, and it would simply reply with a message such as: 'it was not possible to reach any conclusion'[9]. In our hybrid approach, these hypotheses are passed to the CBR module, together with the findings of the network that indicated each diagnosis. These findings are clustered and used as remindings by the CBR system, ie. in the search for previous cases that have had the same set of findings. A certain number of previous cases is retrieved and a computation of similarity with the new case is carried out, indicating a most similar case. The example below shows how the system would behave according to this description.

   Case presented - AVSD17 : *Left anterior hemi blockage*
   Wide and split S2
   Increased pulmonary flow on CXR
   Hystory of cardiac murmur
   Cardiomegaly on CXR
   Systolic heart murmur
   Frequent respiratory infections
   Growth retardation

   When presented with case AVSD17, the neural network indicates two different hypotheses for the problem: diagnoses AVSD and ASD. Figure 5.15 shows the strongest nodes of the network when activated by case AVSD17.

[9]The system HYCONES, which used the CNM network as the main component of its inference process, decided between two or more hypotheses given by the neural network by analysing the number of punishments stored in each node of the network that was used to produce each hypothesis. The diagnoses indicated by nodes with a higher number of punishments were not considered further. This mechanism has been based on the idea that more specific collections of findings (those represented in nodes with less punishments) are more likely to produce correct answers.



Strongest network nodes reaching AVSD hypothesis

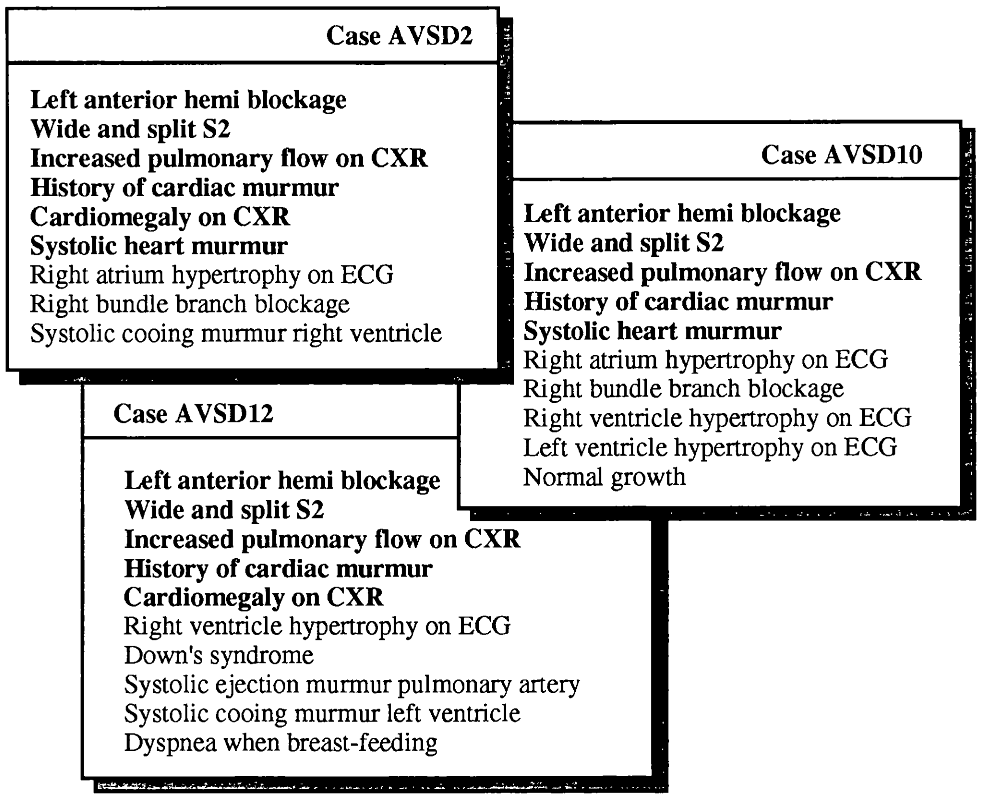
A = Left anterior hemi blockage
B = Increased pulmonary flow on CXR
C = Cardiomegaly on CXR
D = Systolic heart murmur

Strongest network nodes reaching ASD hypothesis

E = Wide and split S2
F = Frequent respiratory infections
B = Increased pulmonary flow on CXR
C = Cardiomegaly on CXR

Figure 5.15: The strongest outputs of the neural network when presented with case AVSD17

The connection that links the node *Left anterior hemi blockage* to AVSD, as well as the connection linking the node *Wide and split S2* to ASD, are the strongest ones. Both are able to indicate the corresponding diagnoses with the highest degree of confidence. To discriminate between the two hypotheses, the findings that activated these nodes (*Wide and split S2* and *Left anterior hemi blockage*) are clustered by the CBR module and used as a reminding to retrieve similar cases. Figure 5.16 shows the cases retrieved by this process.



Case AVSD2

Left anterior hemi blockage
Wide and split S2
Increased pulmonary flow on CXR
History of cardiac murmur
Cardiomegaly on CXR
Systolic heart murmur
Right atrium hypertrophy on ECG
Right bundle branch blockage
Systolic cooing murmur right ventricle

Case AVSD10

Left anterior hemi blockage
Wide and split S2
Increased pulmonary flow on CXR
History of cardiac murmur
Systolic heart murmur
Right atrium hypertrophy on ECG
Right bundle branch blockage
Right ventricle hypertrophy on ECG
Left ventricle hypertrophy on ECG
Normal growth

Case AVSD12

Left anterior hemi blockage
Wide and split S2
Increased pulmonary flow on CXR
History of cardiac murmur
Cardiomegaly on CXR
Right ventricle hypertrophy on ECG
Down's syndrome
Systolic ejection murmur pulmonary artery
Systolic cooing murmur left ventricle
Dyspnea when breast-feeding

Figure 5.16: A set of cases similar to AVSD17 retrieved by the case-matching procedure

The cases AVSD2, AVSD10 and AVSD12 are retrieved by the created reminding, and AVSD2 is selected as the most similar among the three. In order to reinforce the final result, the problem case is compared with the *diagnosis descriptor* of the corresponding final diagnosis.

3. The third path that may be followed by the reasoning process is characterised by the neural network not being able to find any answer for the given problem. No remindings can be built, and the CBR module has to go through all the cases in the library looking for a close match. The final answer has then to be reinforced using the *diagnosis descriptors*, as in the examples presented above. Only a very small percentage of cases is diagnosed by this method, as the neural network covers a very large proportion of the problem space and can usually give an answer, even if with a low confidence degree. Cases that cannot be diagnosed by the network are usually very atypical and therefore difficult to diagnose. This will be discussed further in chapter 6.

After arriving at a final solution, the system has to verify that it is credible. This is important here especially because we have disregarded the minimum degree of confidence accepted by the neural network (we have considered hypotheses coming from the network whether they carry a high or low degree of confidence). Therefore, an additional mechanism has to be used in order to give credibility to a solution found by the hybrid NN-CBR system. This is presented in the next section.

5.9 Reinforcing a Final Result

The method used in the hybrid NN-CBR system to reinforce a final result starts by computing the average intra-class similarity for all cases in the case library taking into account only the attributes *trigger* and *primary findings* of the *diagnosis descriptors*. The similarity between a case C_j and a *diagnosis descriptor* D_k is computed through the formula:

$$Sim(Cj, Dk) = \frac{\alpha \times \sum_{i=1}^{n} \beta \times Spec(Fi, Dk) + Sens(Fi, Dk)}{\sqrt{N_{findings}}}$$

where i ranges through the n common findings between the case considered and *diagnosis descriptor* D_k (*trigger* and *primary findings* only), $Spec(F_i,D_k)$ is the specificity of finding F_i for the diagnosis D_k, and $Sens(F_i,D_k)$ is the sensitivity of finding F_i for the same diagnosis. The constant β is used to attribute a higher importance to the specificity of a finding in relation to its sensitivity. This is justified by the fact that the sensitivity of a finding only measures how frequent the finding is in a diagnosis. It does not show how discriminating the finding is, ie. the efficiency with which the finding can be be used to discriminate among diagnoses. The specificity degree is a factor that is more effective for discrimination as it accounts for the number of times that a finding appears for one diagnosis but not for others. The variable α is used to give an increasing degree of importance for findings that have been classified as *primary* and *triggers*, respectively. Finally, the variable $N_{findings}$ corresponds to the number of findings used to describe the problem case.

The sensitivity degree of a finding in relation to a diagnosis measures its frequency for that diagnosis. It can be obtained through the formula (Owens et al 1990):

$$Sens(F,D) = n_{DF}/n_D$$

where n_{DF} is the number of cases with diagnosis D where finding F is present and n_D is the total number of cases with diagnosis D.

The specificity degree $Spec(F,D)$ of a finding F in relation to a diagnosis D is inversely proportional to the the frequency with which a finding appears in diagnoses other than D. *The* specificity degree of a finding can be obtained through the formula (Owens et al 1990):

$$Spec(F,D) = n_{wFD}/n_{wD}$$

where n_{wFD} is the number of cases without finding F and with other diagnoses than D, and n_{wD} is the total number of cases with diagnoses other than D.

Having determined the average intra-class similarity for each diagnosis considered, we must then calculate the intra-class similarity between the problem case and the diagnosis indicated by the system. The same formula for the computation of the similarity between a case C_j and a *diagnosis descriptor* D_k is used, with the only difference that here all the attributes of the diagnosis descriptor are considered. Using all the attributes at this stage enables cases that do not have a sufficient number of important attributes (*trigger* and *primary findings*) to still have a minimum level of similarity with the *diagnosis descriptor*.

To exemplify this procedure, let us continue with the diagnosis of case AVSD17. AVSD has been identified by the NN-CBR system as the hypothesis for the case. To reinforce this hypothesis, AVSD will be compared with the diagnosis descriptor of AVSD. The average intra-class similarity of AVSD has been computed previously as 1.8, considering only *triggers* and *primary findings* of AVSD. The calculation of the intra-class similarity of the case AVSD17 is shown below (this time taking into consideration the *trigger*, *primary findings* and *secondary findings* of AVSD):

Common findings:

trigger: *Left anterior hemi blockage*, spec: 0.98, sens: 0.80, $\alpha = 1.0$

primary: *Increased pulmonary flow on CXR*, spec: 0.47, sens: 0.87, $\alpha = 0.5$

Cardiomegaly on CXR, spec: 0.57, sens: 0.73

secondary: *Frequent respiratory infections*, spec: 0.76, sens: 0.27, $\alpha = 0.25$

Systolic heart murmur, spec: 0.76, sens: 0.33

Hystory of cardiac murmur, spec: 0.59, sens: 0.67

Wide and split S2,spec: 0.79, sens: 0.27

For $\beta = 2.0$, the degree of similarity between AVSD and AVSD17 is 2.27.

As the degree of similarity between AVSD17 and AVSD (2.27) is bigger than the average intra-class-similarity calculated previously for AVSD (1.8), AVSD is considered to be a credible result for the problem.

After confirming the result, an explanation is built using the retrieved previous case and the corresponding *diagnosis descriptor*. This explanation highlights the importance of the findings present in the problem case, according to their characterisation in the descriptor of the final diagnosis. Furthermore, the matches and mismatches of the problem case with similar previous cases are used to support the answer given to the user. The system produces the following explanation for the AVSD17 case:

"The finding *Left anterior hemi blockage* has been identified as a *trigger* of the diagnosis AVSD. Thus, this finding indicates the diagnosis AVSD with a high degree of confidence. The findings *Increased pulmonary flow on CXR* and *Cardiomegaly on CXR* were identified as *Primary findings* of AVSD cases, and as such they reinforce this diagnosis with a high degree of confidence. The findings *History of cardiac murmur* and *Frequent respiratory infections* support the same diagnostic hypothesis. Although *Wide and split S2* is a *trigger* of another diagnosis (ASD), it can also be observed in some AVSD cases. Case AVSD2 is the most similar previous case in the library, and it also carries both the AVSD *trigger Left anterior hemi blockage* and the ASD *trigger Wide and split S2*. The following findings are common among the two cases: *Left anterior hemi blockage*, *Wide and split S2, Increased pulmonary flow on CXR, History of cardiac murmur, Cardiomegaly on CXR* and *Systolic heart murmur*. The finding *Growth retardation* of the problem case was not matched. The findings *Mid-diastolic rumbling left ventrivle, Right atrium hypertrophy on ECG, Systolic cooing murmur right ventricle* and *Right bundle branch block* of the retrieved case were not matched.

However, there is a degree of similarity sufficiently high between the problem case and the AVSD *diagnosis descriptor* to indicate this diagnosis as a final result.''

5.10 Final Remarks

In summary, the complete reasoning process of the hybrid NN-CBR system can be described by the four '*REs*' steps:

- *REtrieve:* the neural network proposes solutions and provides remindings that are used for the retrieval of similar past cases in the case library;
- *REuse*[10]*:* the solution of the most similar case is reused for the new case. For the types of problems that we target, this step consists simply in classifying the new case in the same way as the retrieved past case has been classified;
- *REvise:* the degree of credibility of the solution is computed using the knowledge stored in the diagnosis descriptors (solutions with low levels of credibility are not considered);
- *REtain:* The new case is stored in the case library and later used to train the neural network.

The last step of the problem-solving procedure consists of building an explanation for the solution found, by using the retrieved case and the corresponding *diagnosis descriptor.* Relying on causal knowledge (*diagnosis descriptors*) as well as on previous experiences to build explanations has proved to be useful in improving the level of acceptance of the explanations provided by expert systems (Georgin et al. 1994). From a cognitive point of view, using both general and specific knowledge is more plausible than relying solely on some type of compiled knowledge. There is no shortage of confirmations from psychology (e.g. see Estes 1984; Sternberg1984) that people keep in their memories experiences represented in single episodes, as well as summary representations of these experiences. Both types of knowledge are used to understand and solve a newly-posed problem. NN-CBR also stores

[10]The *Reuse* step applies mostly to systems that deal with design and planning tasks, where a complex previous solution has to be modified according to requirements and constraints of the new case.

single cases and summary representations of these cases (*diagnosis descriptors* and neural networks), and attempts to use the two types of knowledge in reasoning.

The hybrid NN-CBR architecture also proposes solutions for problems in the area of hybrid connectionis/symbolic systems, and CBR. For instance, the difficulty in understanding the knowledge stored in a neural network has been scaled down by the creation of *diagnosis descriptors* that interpret the connections and weights of the network in three attributes: *triggers, primary* and *secondary findings.* Another problem addressed has been that of referring to relevant previous experiences in case-based systems, which we have tackled by using an alternative indexing scheme based on information that can be obtained from the neural network.

The next chapter presents validation results obtained from the application of the methods presented here in the domain of cardiology, as well as in soybean disease diagnosis, mushroom classification, and in solving the so-called MONK's Problems.

Chapter 6

Validation Results

The hybrid NN-CBR approach has been tested firstly in the Congenital Heart Disease (CHD) domain. Two different databases provided by the Institute of Cardiology, RS (Porto Alegre, Brazil) were used to train and test the overall software. The experiments carried out are detailed in the next sections. The hybrid approach has been also tested in other domains, namely mushroom classification, soybean disease diagnosis and solving the MONK's problems (Thrun 1991). The main objective of this further evaluation has not been to show that the NN-CBR performance is superior to that of other systems originally designed for each application. Instead, our main goal has been to show that the NN-CBR model is also suitable for applications other than the diagnosis of CHD. Furthermore, we wanted to show that, for other applications, the performance of the NN-CBR system would again be superior to that of the CNM network on its own.

6.1 Diagnosing three Frequent CHD Problems

The hybrid NN-CBR model presented in the previous chapter has been evaluated first in the domain of cardiology using a database obtained from the records of the Institute of Cardiology in the Brazilian state of Rio Grande do Sul. This database contained the three most frequent isolated CHD diagnoses, namely *Atrial septal defect* (ASD), *Ventricular septal defect* (VSD) and *Atrioventricular septal defect* (AVSD). These heart diseases can be grouped together because they usually present the symptom *Increased pulmonary and flow on CXR.* A total of 99 cases for these three different Congenital Heart Diseases (CHD) can be found in this database. The distribution of the cases for each diagnosis is the following: 24 AVSD cases, 40 VSD cases and 35 ASD cases. This database will be called *Chd3*.

The cases for these three diseases have been collected with the supervision of an expert in CHD. The patient's history, physical examination, CXR and ECG findings were also extracted from the medical files and stored in the *domain knowledge* hierarchy. The system was trained with two-thirds of the cases randomly selected from this database, and tested with the remaining one-third of cases. The performance of NN-CBR was compared with that of the CNM and the nearest-neighbour algorithm. Table 6.1 shows the results of this basic step of validation.

Results	**NN-CBR**	**CNM**	**nearest-neighbour**
Correct	31 93.94%	28 84.85%	26 78.79%
Misclassified	0 0.00%	1 3.03%	5 15.15%
No conclusions	2 6.06%	4 12.12%	2 6.06%
Total	33 100.0%	33 100.0%	33 100.0%

Table 6.1: CHD, training-set A, testing-set A

The performance of NN-CBR is slightly better than that of the CNM. This happens mainly because for most of the problems that the neural network cannot solve, the CBR module can find a credible answer. NN-CBR also achieves a higher level of accuracy than the nearest-neighbour algorithm used by our CBR module. This can be explained by the fact that the main reasoning mechanism used by NN-CBR is the CNM. In circumstances where the CNM has a better performance than the nearest-neighbour method, the NN-CBR system will also perform better.

Two other symbolic learning algorithms have also been tested with database *Chd3* (in Reategui, Campbell & Leao (1996a)), namely Pbl3 (Uehara et al. 1993) and ID3[1] (Quinlan 1986). Pbl3 (Prototype-based Learning) is an inductive algorithm capable of learning concept descriptions, consisting of prototypical attributes and attribute importances, by using a distance metric based on prototype and information theory. Pbl3 answered correctly 29 out of the 33 CHD cases (87.88%), reaching a performance close to that of NN-CBR. However, the inference approach developed in the NN-CBR presents some advantages over Pbl3. In NN-CBR, the search for a best match in the case library is narrowed via the use of the neural network. Pbl3 inspects all the cases in the library during the categorisation process, which can be expensive for large case libraries.

NN-CBR also showed a better performance than the decision trees generated through the simplified version of ID3. The decision trees answered correctly 25 out of the 33 CHD cases (75.75%), while 31 correct answers were given by NN-CBR. Another advantage that favours the reasoning mechanism employed by NN-CBR is that the discrimination trees generated by ID3 can be complex and difficult to understand. A pruning procedure may be used to simplify the trees and thus improve their comprehensibility. However, even after pruning a tree may be too big to be comprehensible (Feng & Michie 1994). An alternative solution is to convert the trees into rules, which have the same expressive power, but which seem to lend themselves to more user-friendliness (Quinlan 1993). However, the problem of comprehensibility still holds at some level of intensity in rule-based systems, and because of this it is recommended that induced rules be as short as possible. To induce short rules, one must usually relax the requirement that the induced rules be consistent with all the training data. Such measures may encounter a complexity barrier that limits how much can be done in order to make a rule-set comprehensible, without letting it lose too much of its accuracy.

A *diagnosis descriptor* compacts what would be a large set of rules into a simple record-like structure, which describes only the importance of findings for a diagnosis, without

[1]A simplified version of the ID3 algorithm which does not deal with unknown attribute values was used here. The reason for this choice was that the description of the cases in the training set was not given in terms of attributes/values, but simply as lists of medical findings, so that no difference between unknown and absent attributes was made visible.

enforcing the 'all-or-nothing' conditions found in rules. The example below (figure 6.1) shows how a *diagnosis descriptor* summarises the knowledge of a set of rules for the diagnosis of AVSD.

If *Left anterior hemi blockage*
  Then AVSD
If *Left anterior hemi blockage*
  Down's syndrome
  Then AVSD
If *Left anterior hemi blockage*
  Increased pulmonary flow on CXR
  Then AVSD
If *Left anterior hemi blockage*
  Cardiomegaly on CXR
  Then AVSD
If *Left anterior hemi blockage*
  Right ventricle hypertrophy
  Then AVSD

If *Left anterior hemi blockage*
  History of cardiac murmur
  Then AVSD
If *Left anterior hemi blockage*
  Left ventricle hypertrophy
  Then AVSD
If *Down's syndrome*
  Increased pulmonary flow on CXR
  Then AVSD
If *Down's syndrome*
  Tachypnea
  Then AVSD

Diagnosis descriptor ▶ **AVSD**

Trigger: Left anterior hemi blockage

Primary: Down's syndrome
Increased pulmunary flow on CXR
Cardiomegaly on CXR

Secondary: Right ventricle hypertrophy
Hystory of cardiac murmur
Left ventricle hypertrophy
Tachypnea

Figure 6.1: A set of symbolic rules and a *diagnosis descriptor* representing the AVSD problem

When designing the *diagnosis descriptors*, our goal has not been to define logical operators to determine the possible combinations of attributes that would be necessary for the identification of a diagnosis, e.g. that the presence of the *trigger* with two or more *primary* *findings* would be sufficient to determine a particular diagnosis. The number of combinations of findings that can lead to the identification of a diagnosis is usually very large, and trying to describe all these combinations explicitly in a symbolic way would result in a representation that would not be easy to read. The examples given in chapter 5 (section 5.5), as well as the example above, show how a small number of combinations of evidence is expanded into a set of rules that is not very easy to comprehend without a detailed examination. The solution we have created for this problem has been to keep a representation of the combinations of findings in the neural network, and to define only the importance of the findings in the *diagnosis descriptors*. This form of representation allows the *diagnosis descriptors* to have a simpler structure which is easy to read and understand. Another property enabling the simplification of the *diagnosis descriptors* is that they are not used for reasoning purposes, but more for consultation and explanation purposes. In the NN-CBR approach, the mechanism that determines the sufficiency of evidence for a hypothesis to be singled out is the neural network.

A second experiment in the CHD domain has been the presentation of 38 other cases from four different CHD problems to the systems for diagnosis: *Aortic Stenosis* (AST), *Aortic Coarctation* (ACOA), *Persistent ductus arteriosis* (PDA), *Tetralogy of Fallot* (TFLT). The expected result was that the systems would not be able to diagnose any of the cases. Table 6.2 shows the results for this experiment.

Results	**NN-CBR**	**CNM**	**nearest-neighbour**
Misclassified	9 23.68%	30 78.95%	32 84.21%
No conclusions	29 76.32%	8 21.05%	6 15.79%
Total	38 100.0%	38 100.0%	38 100.0%

Table 6.2: Systems trained to diagnose ASD, VSD and AVSD, diagnosing four other diseases: AST, ACOA, PDA and TFLT

In this experiment, NN-CBR performs considerably better than both the CNM and the nearest-neighbour algorithm. This is mainly because the NN-CBR system has a verification method that calculates and enforces a minimum level of credibility for final results. This method, presented in detail in chapter 5, computes the average intra-class similarity for all cases in the case library taking into account the attributes *trigger* and *primary findings* of the *diagnosis descriptors.* A diagnostic hypothesis is refuted if its similarity with the problem case is smaller than the average intra-class similarity previously computed for the same diagnosis. The threshold mechanism used by the CNM[2] is not as efficient when presented with cases that the neural network should not be able to diagnose. The neural network always seems to find some evidence indicating one diagnosis or another. Regarding the nearest-neighbour algorithm used here, it does not include any control on the minimum similarity degree required for the system to present an answer. Because of this, the nearest-neighbour method diagnosed inappropriately almost all the cases in this last experiment. However, it would be possible to include in the nearest-neighbour method a procedure that would only accept answers carrying a confidence degree higher than a given threshold. The inclusion of such a control would improve the performance of the nearest-neighbour method in relation to the identification of cases that do not belong to any of the classes that the system was taught to identify.

6.2 CHD - Diagnosis of four other Problems

The 38 cases used in the last experiment (cases with heart diseases AST, ACOA, PDA and TFLT) are in fact part of a second database provided by the same institute (Institute of Cardiology - Rio Grande do Sul), which contains a total of 115 cases distributed in the following way: 28 AST cases, 28 ACOA cases, 31 PDA cases and 28 TFLT cases. These cases have been collected without expert supervision and present incomplete descriptions.

This database has been combined with the *Chd3* database to train and test a system for the diagnosis of all seven types of CHD problems (the database containing cases with the

[2]In the training period of the CNM network, all the weights of the network are calculated using the confidence threshold *(Tacc)* as a reference (see further details in Appendix B). When solving new problems, this threshold is used to determine whether an answer can be passed forward or not.

seven diseases will be called *Chd7*). The total number of cases of *Chd7* was 214. Table 6.3 shows the performance of NN-CBR, the CNM and the nearest-neighbour algorithm when diagnosing a set of cases extracted from *Chd7*:

Results	**NN-CBR**	**CNM**	**nearest-neighbour**
Correct	52 73.24%	42 59.15%	36 50.7%
Misclassified	9 12.68%	10 12.12%	27 38.0%
No conclusions	10 14.08%	19 26.76%	8 11.3%
Total	71 100%	71 100%	71 100%

Table 6.3: The performance of the systems when diagnosing the seven CHD diseases

Similar conclusions to those of the first experiment can be drawn from the results shown in table 6.3. The NN-CBR system again shows a better performance than that of the CNM and the nearest-neighbour algorithm. However, it is noticeable that the overall performance of each system is degraded in relation to its performance in the first experiment (table 6.1). There is a very important factor that has induced this drop in performance. The cases of database *Chd7* have been transcribed from old medical records to our case format without the supervision of an expert. The transcription of a case from an old record to a different format consists of two main tasks: understanding the problem sometimes explained in a somewhat free format, and then restating the problem using a particular vocabulary. VanLehn (1989) reports several experiments showing that experts have a better capacity of understanding and retaining details of a newly-posed problem. In our CHD experiment, this claim is confirmed by the fact that the cases collected with expert supervision had more detailed descriptions. While in database *Chd3* an average of 9 medical findings was used to describe each case, for database *Chd7* an average of 5 findings was observed. To some extent, it can be claimed that the knowledge of an expert was embedded in the cases of database *Chd3*, while the knowledge of a less experienced cardiologist was embedded in the cases of database *Chd7*. Therefore, NN-CBR was able to learn more 'effective rules' and thus perform better using the cases of database *Chd3*.

Machado, Rocha & Leao (1990) also report the differences between expert and non-expert knowledge. They say that despite using smaller graphs, the experts displayed a larger number of pathways capable of associating evidence with the diagnostic hypotheses, evidencing a greater semantic capacity in comparison to non-experts. The same paper also reports that non-experts were frequently unable to include the key items of evidence employed by experts in their reasoning, even using large clusters (ie. a combination of a large number of items of evidence). These statements support the results obtained in our research, where the performance of each of our trial systems using the cases transcribed with the supervision of an expert had a better performance than the same systems using cases collected without expert supervision. Table 6.4 shows the performance of NN-CBR for each of the seven types of CHD diagnoses separately.

	ASD	VSD	AVSD	AST	ACOA	TFLT	PDA	*Total*
Correct	13 100.0%	10 90.9%	8 88.9%	6 66.7%	6 66.7%	5 55.5%	4 44.4%	52 73.2%
Misclassified	0 0%	0 0%	0 0%	1 11.1%	2 22.2%	2 22.2%	4 44.4%	9 12.7%
No conclusions	0 0.0%	1 9.1%	1 11.1%	2 22.2%	1 11.1%	3 33.3%	2 22.2%	10 14.0%
Total	13 100%	11 100%	9 100%	9 100%	9 100%	9 100%	11 100%	71 100%

Table 6.4: The performance of NN-CBR for each of the seven types of CHD diagnoses

The results demonstrate that NN-CBR was able to solve more accurately problems related to the first three CHD diagnoses (ASD, VSD and AVSD), ie. the diagnoses for which the cases were transcribed with the supervision of an expert. The *diagnosis descriptors* built for these three diagnoses are also more similar to the 'mean' knowledge graphs elicited from multiple experts. Figures 6.2 to 6.8 show the leftmost trees of the experts' and non-experts' knowledge graphs, and the *trigger* and *primary findings* of the *diagnosis descriptors* for each diagnosis. A comment of Dr. Beatriz Leao about the results obtained is also presented for each of the problems addressed.

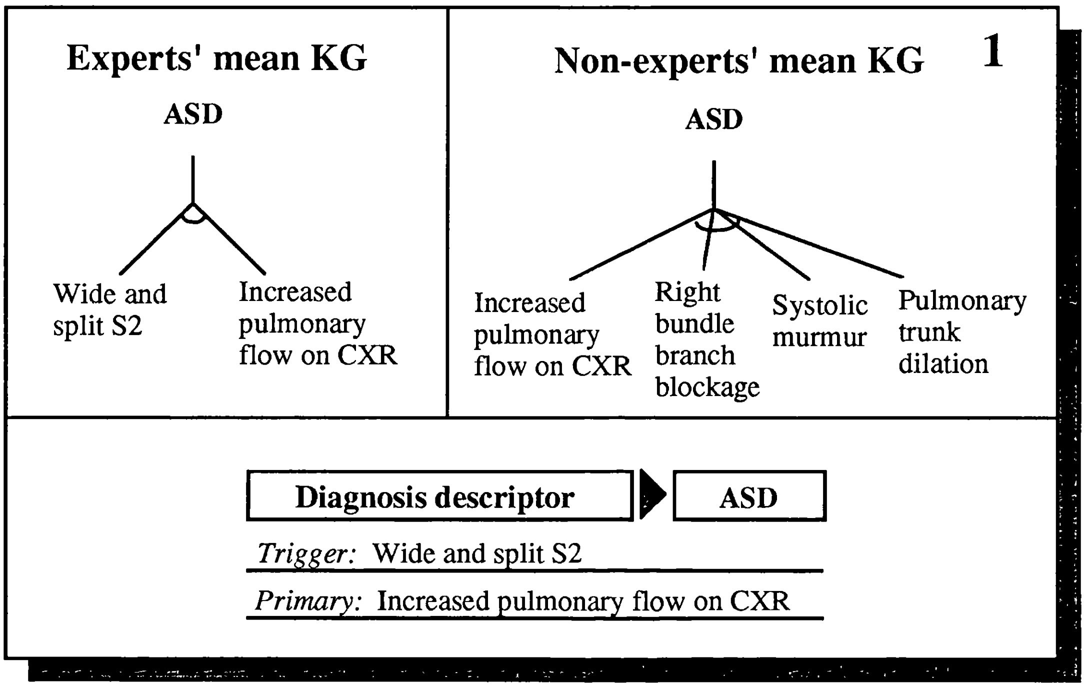

Experts' mean KG
ASD
Wide and split S2
Increased pulmonary flow on CXR
Non-experts' mean KG
1
ASD
Increased pulmonary flow on CXR
Right bundle branch blockage
Systolic murmur
Pulmonary trunk dilation
Diagnosis descriptor
ASD
Trigger: Wide and split S2
Primary: Increased pulmonary flow on CXR

Figure 6.2: Comparing expert and non-expert KGs with the *diagnosis descriptor* of ASD

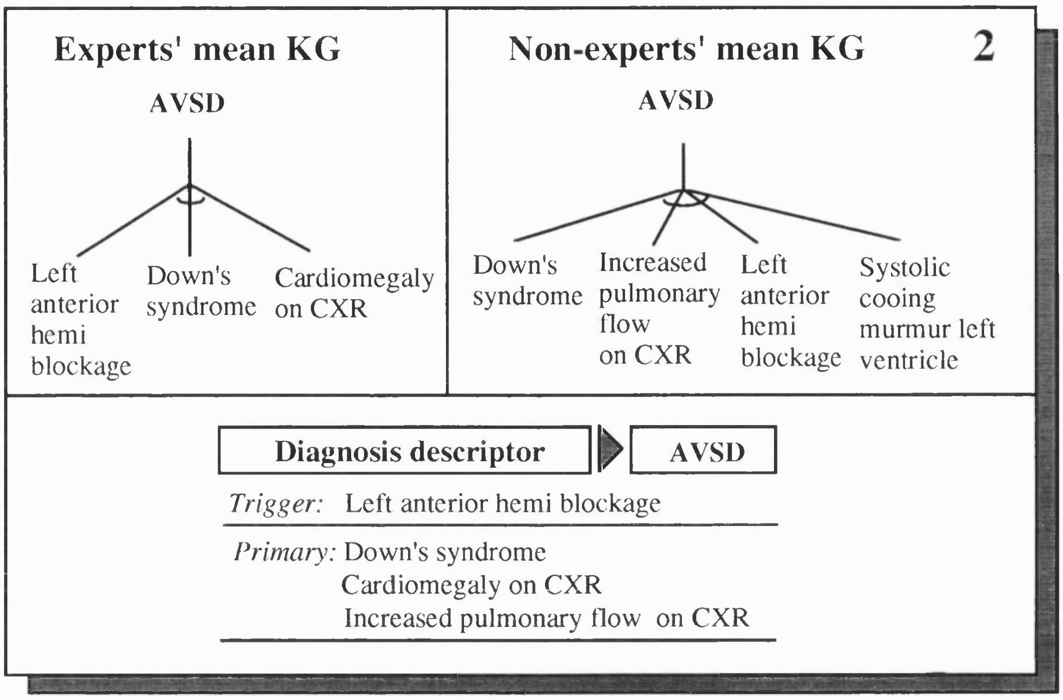

Experts' mean KG
AVSD
Left anterior hemi blockage
Down's syndrome
Cardiomegaly on CXR
Non-experts' mean KG
2
AVSD
Down's syndrome
Increased pulmonary flow on CXR
Left anterior hemi blockage
Systolic cooing murmur left ventricle
Diagnosis descriptor
AVSD
Trigger: Left anterior hemi blockage
Primary: Down's syndrome
Cardiomegaly on CXR
Increased pulmonary flow on CXR

Figure 6.3: Comparing expert and non-expert KGs with the *diagnosis descriptor* of AVSD

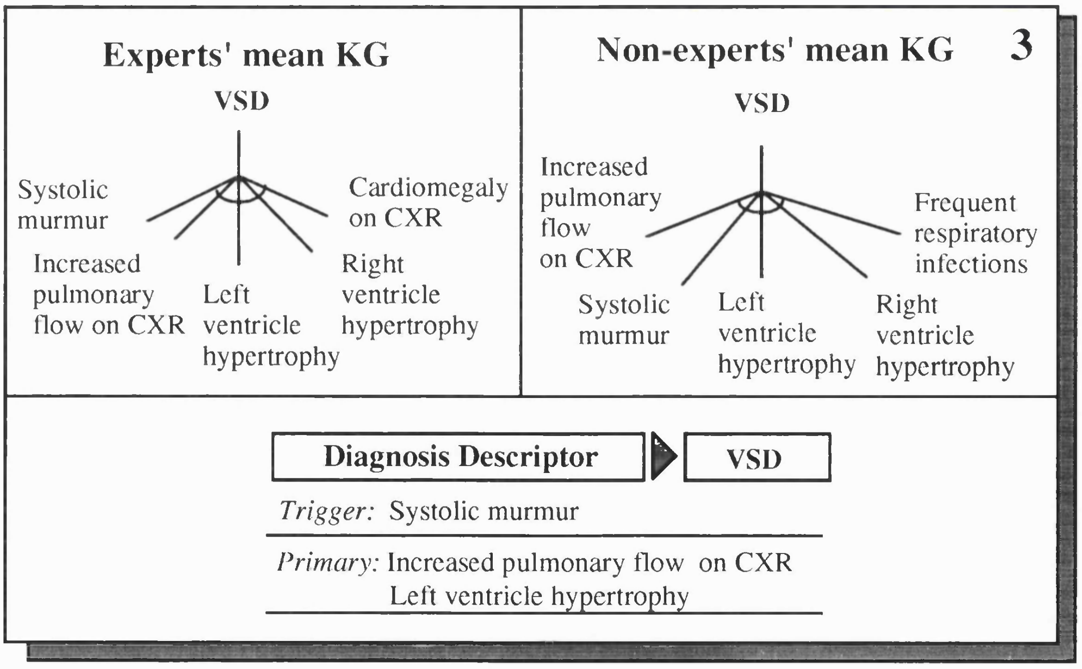

Experts' mean KG
VSD
Systolic murmur
Cardiomegaly on CXR
Increased pulmonary flow on CXR
Left ventricle hypertrophy
Right ventricle hypertrophy
Non-experts' mean KG
3
VSD
Increased pulmonary flow on CXR
Frequent respiratory infections
Systolic murmur
Left ventricle hypertrophy
Right ventricle hypertrophy
Diagnosis Descriptor
VSD
Trigger: Systolic murmur
Primary: Increased pulmonary flow on CXR
Left ventricle hypertrophy

Figure 6.4: Comparing expert and non-expert KGs with the *diagnosis descriptor* of VSD

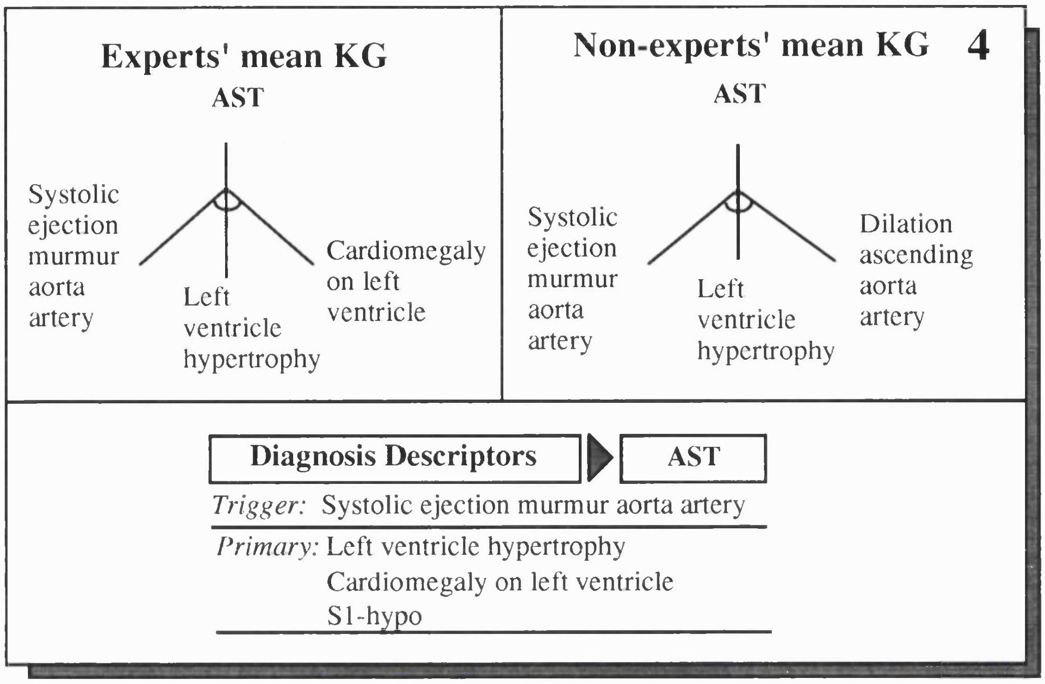

Experts' mean KG
AST
Systolic ejection murmur aorta artery
Left ventricle hypertrophy
Cardiomegaly on left ventricle
Non-experts' mean KG
4
AST
Systolic ejection murmur aorta artery
Left ventricle hypertrophy
Dilation ascending aorta artery
Diagnosis Descriptors
AST
Trigger: Systolic ejection murmur aorta artery
Primary: Left ventricle hypertrophy
Cardiomegaly on left ventricle
S1-hypo

Figure 6.5: Comparing expert and non-expert KGs with the *diagnosis descriptor* of AST

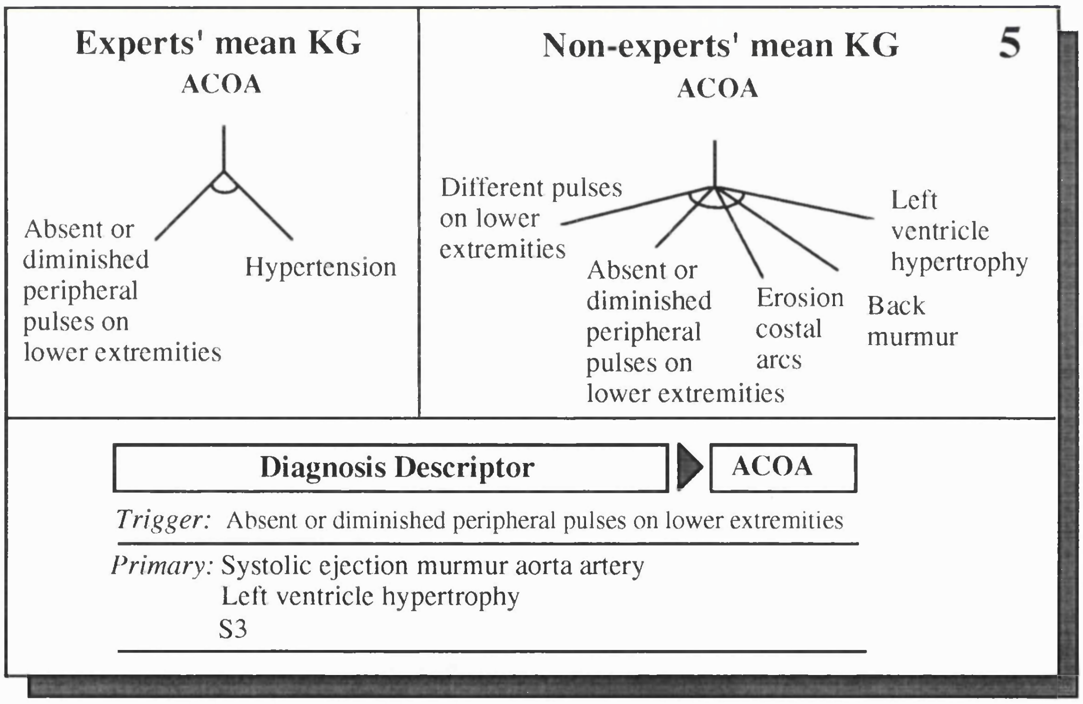

Experts' mean KG
ACOA
Absent or diminished peripheral pulses on lower extremities
Hypertension
Non-experts' mean KG
5
ACOA
Different pulses on lower extremities
Absent or diminished peripheral pulses on lower extremities
Erosion costal arcs
Left ventricle hypertrophy
Back murmur
Diagnosis Descriptor
ACOA
Trigger: Absent or diminished peripheral pulses on lower extremities
Primary: Systolic ejection murmur aorta artery
Left ventricle hypertrophy
S3

Figure 6.6: Comparing expert and non-expert KGs with the *diagnosis descriptor* of ACOA

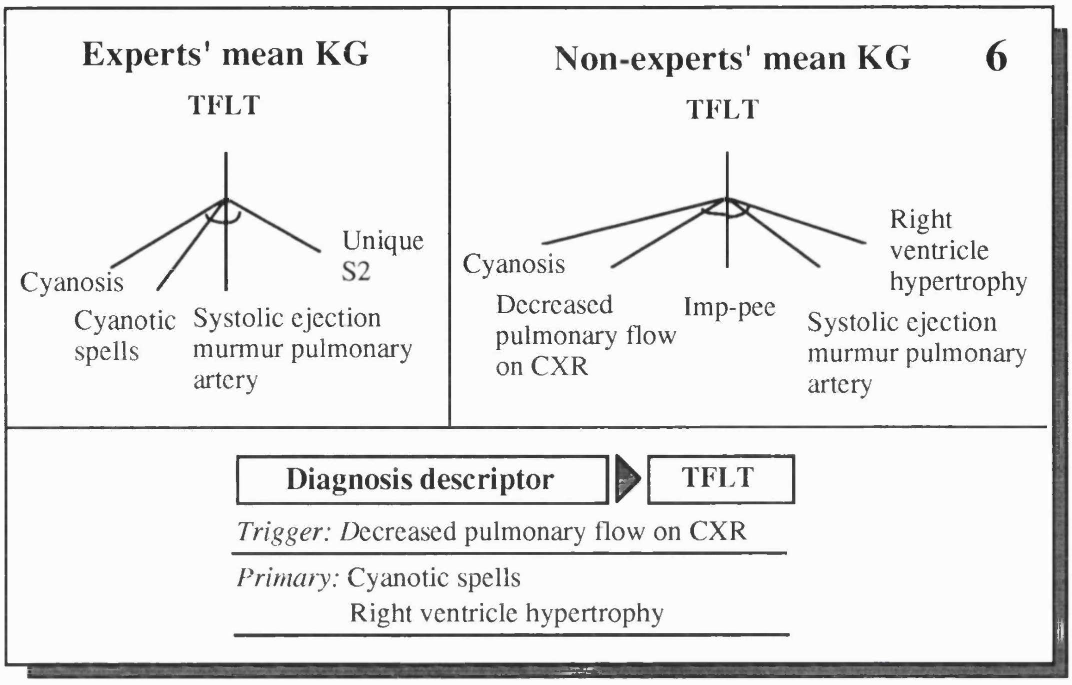

Experts' mean KG
TFLT
Cyanosis
Cyanotic spells
Systolic ejection murmur pulmonary artery
Unique S2
Non-experts' mean KG
6
TFLT
Cyanosis
Decreased pulmonary flow on CXR
Imp-pee
Systolic ejection murmur pulmonary artery
Right ventricle hypertrophy
Diagnosis descriptor
TFLT
Trigger: Decreased pulmonary flow on CXR
Primary: Cyanotic spells
Right ventricle hypertrophy

Figure 6.7: Comparing expert and non-expert KGs with the *diagnosis descriptor* of TFLT

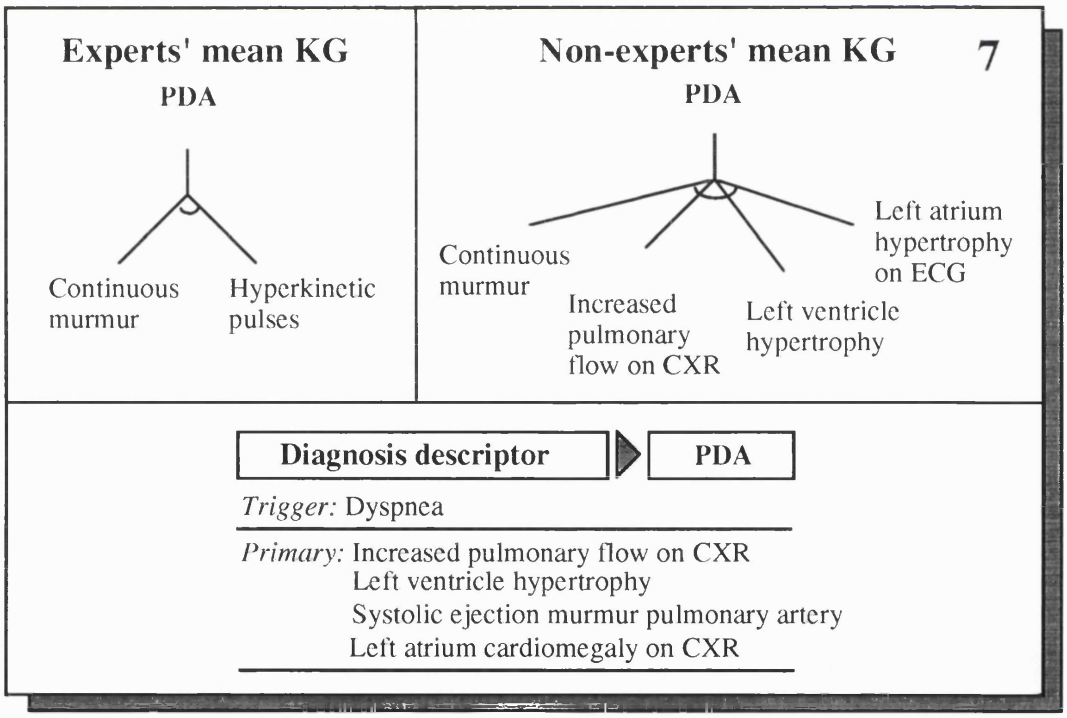

Experts' mean KG
PDA
Continuous murmur
Hyperkinetic pulses
Non-experts' mean KG
7
PDA
Continuous murmur
Increased pulmonary flow on CXR
Left ventricle hypertrophy
Left atrium hypertrophy on ECG
Diagnosis descriptor
PDA
Trigger: Dyspnea
Primary: Increased pulmonary flow on CXR
Left ventricle hypertrophy
Systolic ejection murmur pulmonary artery
Left atrium cardiomegaly on CXR

Figure 6.8: Comparing expert and non-expert KGs with the *diagnosis descriptor* of PDA

Comments of Dr. Beatriz Leao[3] for each diagnosis:

- **ASD, AVSD and VSD:** the classic symptom for ASD is *Wide and split S2*. The other symptoms that appear in the experts' and non-experts' graphs can also be observed in ASD cases, but they are not so specific to this disease (such as *Increased pulmonary flow on CXR*, which is present in all heart diseases with increased pulmonary flow: ASD, VSD, AVSD and PDA). The system was able to detect this difference by classifying *Wide and Split S2* as a *trigger*, and *Increased pulmonay flow on CXR* as a *primary finding*, which is very good. For AVSD and for VSD, the system has also been able to identify the findings that are most specific and frequent.

- **AST** and **ACOA:** The system has been able to identify the most specific findings for each of the two diagnoses (*Systolic ejection murmur aorta artery* for AST and *Absent or diminished peripheral pulses on lower extremities* for ACOA). In AST, the findings selected as *primary* are in accordance with the experts' graphs, with the exception of the finding *S1-hypo*. In ACOA, the systems' description is a bit less accurate, though. The finding *S3*, for instance, is not really relevant for the diagnosis of ACOA. This finding indicates the presence of cardiac insufficiency, which only occurs in the most serious cases of ACOA. But as the records of the patients used to train the system have been taken from hospitalised patients, it is understandable that most of the patients with ACOA would also present the finding *S3*.

- **TLFL:** Although the systems' description for this diagnosis is different from that of the experts' mean knowledge graph, the system has been able to identify important findings in TLFL. These findings appear either in the non-experts' mean knowledge graph (*Decreased pulmonary flow on CXR* and *Right ventricle hypertrophy*) or in the experts' graph (*Cyanotic spells*).

[3]Dr. Leao, who is an expert in CHD, has been responsible for the collection of knowledge graphs from more than 40 experts and non-experts in CHD in an experiment described in Leao (1988) and in Leao & Rocha (1990).

- **PDA:** Here the system has not been able to identify the *trigger* of the diagnosis. The probable reason for this has been that the murmurs which were present in the PDA cases of the training database were classified in different ways (such as *Systolic ejection murmur pulmonary artery*, or *Non-specific systolic murmur*)[4].

Another aspect that has to be taken into consideration here, which justifies minor differences between graphs and *diagnosis descriptors,* is that the graphs shown in figures 6.1 to 6.7 are in fact 'mean' representations of several graphs elicited from multiple experts and non-experts (Leao 1988; Leao & Rocha 1990). That is, for each diagnosis, several graphs elicited from different experts have been merged in one single graph, so as to accumulate the overall knowledge of the population of experts. The methodology used to construct these 'mean' knowledge graphs (Machado, Rocha & Leao 1990) analyses the frequency of occurrence of arcs in the experts' graphs to determine an initial 'population knowledge graph'. The arcs of this graph that have a frequency lower than a given threshold are then pruned, to generate the final mean knowledge graph. As the graphs shown here represent a kind of consensus knowledge of several experts, it is acceptable that the *diagnosis descriptors* do not always have an exact correspondence with what is represented in the mean knowledge graphs. This is especially true if the view of the specialist who collected the cases from which the *diagnosis descriptors* were built diverges from the consensus found in the experts' mean knowledge graph. As observed by Machado, Rocha & Leao (1990), there is often a high degree of divergence in the evidence selected by different specialists to support the same diagnostic hypothesis. Therefore, as the graphs shown here represent the consensus knowledge of several experts, it is acceptable if the *diagnosis descriptors* built according to cases collected by a yet different expert (or non-expert) do not always correspond exactly to what is represented in the mean knowledge graphs.

[4] No PDA case in the database contained a murmur described as *Continuous murmur.* We believe that the system would have a better chance to identify correctly the *trigger* and *primary findings* of PDA if the murmurs appearing in PDA cases were described in a more generic way.

Regarding the percentage of misclassifications, it can be expected to be low for any diagnosis in which the cases we collected with the supervision of an expert. In our experience of the first three types of diagnosis for which our system has been developed, it was exactly 0.0%. For the other four diagnoses, the percentage of misclassifications is higher than these figures. However, it is still much lower than the value of the expression:

100 - *percentage of correct classifications*

which represents the number of incorrect answers that would be presented by the system if no mechanism were used to enforce a minimum level of confidence for an answer to be given to the user. The level of misclassification could be still reduced, though, if cases with more complete and correct representations were introduced into the library and used to refine the knowledge of the system, e.g. somewhat as in Leao & Reategui (1993). In this previous experiment, a neural network created from mean knowledge graphs and whose performance was not satisfactory (42.5% accuracy) was refined using a set of cases. After the training period, the level of accurary presented by the system increased to 78.7%.

In our CHD example, only two correct answers were intercepted by the mechanism that enforces a minimum level of credibility, while seven answers were correctly blocked. In applications where it is desirable that the system should give an answer rather than stating that no conclusion could be reached, the mechanism that checks the credibility of answers can be deactivated. In the CHD example, this would increase the number of correct answers from 52 to 54 (76.1%), but it would also increase the number of misclassifications, from 10 to 17 (23.9%).

The different steps which the reasoning mechanism can follow when diagnosing a case have also been evaluated separately. These steps define three different procedures to solve a given problem:

(1) one hypothesis only is indicated by the neural network. The CBR module has to find a similar previous case to confirm the hypothesis, using remindings provided by the neural network.

(2) two or more hypotheses are made by the neural network. The CBR module uses remindings provided by the neural network to find the most similar previous case

that will reinforce only one of the hypotheses.

(3) no hypothesis is made by the neural network, and the CBR module has to search for a similar previous case without having any clue for the retrieval.

The performance of each of these procedures when diagnosing cases of databases *Chd3* and *Chd7* is presented in table 6.5.

	Chd3 *correct*	*Chd3* *incorrect*	*Chd3* *no concl.*	*Chd7* *correct*	*Chd7* *incorrect*	*Chd7* *no concl.*
(1) One hypothesis given by NN	28	0	2	45	7	9
(2) More than one hypothesis made by NN	3	0	0	6	0	0
(3) No hypothesis made. CBR module works on its own	0	0	0	1	2	1
Total	31	0	2	52	9	10

Table 6.5: The performance of NN-CBR's reasoning procedures for the *Chd3* and *Chd7* databases

In procedure 1, the neural network is the actual mechanism that determines the final answer, as it makes only one hypothesis, and guides the case-based component in the search for similar previous cases supporting this same hypothesis. This method showed a good performance, making no mistakes for the *Chd3* database, and 7 mistakes out of 52 answers for the Chd7 database.

Procedure 2 presented a very high level of accuracy in the diagnosis of problems of both databases (*Chd3* and *Chd7*), demonstrating that it is possible to find correct answers with a good level of confidence for cases that cannot be diagnosed uniquely by the neural network. This procedure uses remindings provided by the neural network to search the case library for similar experiences and discriminate between two or more hypotheses. In both experiments presented in table 6.5, the system reached only correct answers through this method, which demonstrates its capacity in identifying appropriate previous experiences through the use of the neural netowk remindings.

The case-based method diagnosing cases on its own (with no use of remindings coming from the neural network), presented a much lower level of accuracy (procedure 3). The cases left for this procedure are usually more difficult to diagnose, as they do not conform with the norm and, thus, cannot be diagnosed by the 'general rules' represented in the neural network. The idea of leaving such cases to be diagnosed solely by the CBR component comes from Riesbeck & Schank (1989), who say that we only reason from prior experiences when well-established rules are not available to solve the problem. However, there are several difficulties associated with the diagnosis of cases that do not conform with the norm. Firstly, it is hard to determine which parts of these cases can be used as remindings for the retrieval of previous experiences. As these cases are atypical, there is no statistical evidence that the use of such remindings would result in the retrieval of appropriate cases. Secondly, it is difficult to determine how similar a previous case has to be to enable the use of its solution for a new case. Thirdly, but not less importantly, these atypical cases may in fact be the product of a noisy and/or incomplete description of what would instead be more ordinary cases.

We have tried to tackle these problems by using the mechanism that enforces a minimum degree of similarity between a new case and a *diagnosis descriptor*. However, it is certain that the representation of an atypical (or incorrect) case cannot bear too many similarities with the *diagnosis descriptor* for the same diagnosis, as the descriptors take into account mostly the pieces of evidence that are either frequent or specific for a problem (but not for the evidence found to be atypical). Thus, we have had to relax the similarity constraints for procedure 3, so that the system can still find solutions for atypical problems. However a lower level of accuracy should be expected from this procedure. In our architecture, a lower degree of confidence is also given for answers determined by such a method. The advantage of using procedure 3 is that it enables the system to continue to diagnose some cases correctly even when the neural network cannot make any hypothesis or reminding. In applications where the number of misclassifications should be kept as low as possible, this module could be deactivated to guarantee a lower number of errors.

6.3 The Soybean Disease and the Mushroom Classification Database

The main objective of this further validation step has not been to show that the NN-CBR performance is superior to that of other systems originally designed for each application. Instead, our main goal has been to show that the NN-CBR model is also suitable for applications other than the diagnosis of CHD. Furthermore, we wanted to show that for other applications, the performance of the NN-CBR system would again be superior to that of the CNM network on its own.

The *soybean disease* and the *mushroom* databases have been obtained from Murphy & Aha (1994). The *soybean* database contains 630 cases of 15 different diagnoses (Michalsky & Chilausky 1980), while the mushroom database is composed of 8400 cases belonging to one of two classes. One-fifth of the cases from the *mushroom* database were randomly selected to train and test the CNM network and NN-CBR (ie. 1624 cases, where 812 were used in training, and another 812 were used in testing the system). The results of the tests for both databases are presented in table 6.6.

Results	**mushroom NN-CBR**		**mushroom CNM**		**soybean NN-CBR**		**soybean CNM**	
Correct	803	98.9%	774	95.3%	310	91.2%	269	87.0%
Misclassified	9	1.1%	9	1.1%	24	7.0%	21	6.2%
No conclusions	0	0.0%	29	3.6%	6	1.8%	23	6.8%
Total	812	100%	812	100%	340	100%	340	100%

Table 6.6: The performance of the CNM network and NN-CBR in the diagnosis of soybean diseases and in the classification of mushrooms

NN-CBR and the CNM network achieve a high level accuracy in the *mushroom* and the *soybean* example. NN-CBR also shows a superior performance to that of the CNM in both problems. It is easy to observe that the cases diagnosed by the CNM where there were no conclusion, are diagnosed correcly by the NN-CBR in most circumstances. Table 6.7 shows the performance of each procedure involved in the reasoning process for the two databases (mushroom and soybean diseases).

	mushroom correct	*mushroom incorrect*	*mushroom no concl.*	*soybean correct*	*soybean incorrect*	*soybean no concl.*
(1) One hypothesis given by NN	774	9	0	296	16	5
(2) More than one hypothesis made by NN	29	0	0	11	0	0
(3) No hypothesis made. CBR module works on its own	0	0	0	3	8	1
Total	803	9	0	310	24	6

Table 6.7: The performance of each procedure in the reasoning process for the *soybean* and the *mushroom* databases

The efficiency of the indexing mechanism that uses remindings coming from the neural network to select similar previous cases is again confirmed in table 6.7 (procedures 1 and 2). In the *mushroom* example, only 9 answers (out of 783) were given incorrectly by procedure 1. The same procedure generated only 16 incorrect answers (out of 312) for the *soybean* problem.

In the *mushroom* example, as well as in the *soybean* example, all the answers provided by procedure 2 have been correct. The same level of accuracy can be observed in the CHD example, which confirms the efficiency of this indexing mechanism in identifying relevant previous experiences for the discrimination among two or more hypotheses.

The case-based method used to diagnose more atypical cases on its own (3) shows a much lower level of accuracy. However, as discussed previously, this third procedure is left with the cases that are the most difficult to diagnose, and thus a lower level of accuracy should be expected from it. The goal of using such a method is that the diagnosis of complicated cases can still be attempted, even if a lower level of confidence is attached. But we stress again that this mechanism should only be activated for applications where giving an answer is better than not reaching any conclusion, and that it can be deactivated for applications where the level of misclassifications has to be kept as low as possible.

The mechanism that intercepts answers that are believed not to be credible also showed a good performance. In the mushroom problem, no answer has been intercepted incorrecly. In the soybean problem, five answers were intercepted correctly, and no correct answer was blocked (the 6th answer for which the system could not conclude anything was not an intercepted answer, but simply a case that neither the NN not the CBR module could diagnose).

6.4 The MONK's Problems

The MONK's Problems rely on an artificial robot domain, in which robots are described by six different attributes (Thrun 1991):

- head-shape (A): round (A1), square (A2), octagonal (A3)
- body-shape (B): round (B1) , square (B2), octagonal (B3)
- is-smiling (C): yes (C1), no (C2)
- holding (D) : sword (D1), balloon (D2), flag (D3)
- jacket-colour (E): red (E1), yellow (E2), green (E3), blue (E4)
- has-tie (F): yes (F1), no (F2)

The learning task is a binary classification task. Robots either belong to a certain class (X) or not. But instead of providing a complete class description for the learning problem, only a subset of all 432 possible robots with their classification is given. The learning task is then to generalise over these examples and, if possible, to derive a simple class description. Table 6.8 shows the performance of NN-CBR for the 3 MONK's Problems.

Results	**MONK's Problem 1**		**MONK's Problem 2**		**MONK's Problem 3**	
Correctly diagnosed	354	81.95%	261	60.4%	374	86.6%
Misclassifications	48	11.1%	156	36.1%	48	11.1%
No conclusions	30	6.95%	15	3.5%	10	2.3%
Total	432	100.0%	432	100.0%	432	100.0%

Table 6.8: The performance of NN-CBR for the MONK's Problems 1, 2 and 3

The system had a satisfactory performance for MONK's Problems 1 and 3. For the second problem, the system did not manage to learn the classification rules well, which led to a poor performance.

In the first MONK's Problem, the target rule used to build the database (which learning algorithms should ideally learn) was:

'If [jacket-colour = red (E1)] OR [head-shape = body-shape (A=B)], then the robot belongs to class X, otherwise it does not'

This rule is composed of two clauses, each of which requires a different type of reasoning:

1. determining which feature/value attributes are important for the identification of a class (eg. E1 indicates class X)

2. determining that certain relationships among attributes can serve as evidence for the identification of a class (eg. in class X, head-shape and body-shape are the same).

The type of reasoning of item 1 can be reproduced well by our hybrid model of reasoning. For instance, NN-CBR selected E1 as the trigger of class X. Regarding the rule stated in item 2, NN-CBR does not have the means to represent explicitly this kind of relationships between attributes. However, the CNM network kept nodes with 'same body and head shapes' with very high connection weights: (A1, B1), (A2, B2) and (A3, B3). This enabled the system to perform well, nevertheless, for the first MONK's Problem.

In the second MONK's Problem, the target rule used to build the database was:

'When exactly two of the six attributes have their first value, the class is X'

This second problem requires the representation of a type of classificatory knowledge that could not be learned at all by our hybrid approach. NN-CBR does not consider the order of the values of each attribute as evidence for the diagnostic process. Instead, the system relied on the importance of feature/value attributes for each class, which led to a bad performance.

In the third MONK's Problem, the target rule used to build the database was:

'When [jacket-colour = green (E3) AND holding = sword (D1)] OR [jacket-colour not blue (E1 or E2 or E3) AND body-shape not octagonal (B1 or B2)], the class is X'

In the third MONK's Problem, once again each clause of the rule requires a different type of reasoning. The first clause requires the representation of the importance of features in relation to a diagnosis, as in item 1 of the first MONK's Problem. This knowledge was not represented explicitly in the diagnosis descriptors. It was only kept in the neural network in the form of a node clustering the findings E3 and D1. The weight of this node was not very high, though, as 5% misclassifications have been introduced intentionally in the training set as to test the capability of learning algorithms to cope with noise. In the training set, the two findings (E3 and D1) appeared together in examples of class X and examples of 'not X'.

The second clause uses conjunctions of negative statements, which cannot be represented explicitly in NN-CBR. However, the findings E4 (jacket-colour = blue) and B3 (body-shape = octagonal) were kept in the list of findings of the *diagnosis descriptor* of 'not X'. Furthermore, the neural network kept nodes with high connection weights indicating class X for the clusters: (B1, E1), (B1, E2), (B2, E1), (B2, E2), (B1, E3) and (B2, E3) (representing 'not E4' and 'not B3')[5]. Despite the 5% level of noise intentionally introduced in this database, the system achieved a reasonably good performance as it managed to represent in the neural network and the *diagnosis descriptors* the concepts used to create the database. For all the MONK's Problems, NN-CBR had an equal or better performance than the CNM.

- MONK's Problem 1: 79.6% of correct answers by the CNM;
- MONK's Problem 2: 58.1% of correct answers by the CNM;
- MONK's Problem 3: 86.6% of correct answers by the CNM.

6.5 Further Comments

It has been observed in all experiments mentioned above that there is a substantial reduction of the number of cases considered by the CBR module for each problem solved. In our hybrid approach, differently from the nearest-neighbour algorithm, the number of case comparisons when solving a new problem does not increase linearly with the number of existing cases in the library. The indexing scheme based on the use of knowedge coming from the neural network enables a big reduction in the number of cases compared, as depicted in table 6.9.

	Chd3	*Chd7*	mushroom	soybean	MONK's 1	MONK's 2	MONK's 3
NN-CBR	15	15	300	25	23	43	28
nearest-neighbour	66	143	812	290	124	124	124
reduction	77.3%	89.5%	63.0%	91.4%	81.5%	65.3%	77.4%

Table 6.9: The reduction in the number of case comparisons between the NN-CBR approach and the nearest-neighbour algorithm

[5]The weights of the three last clusters were not very high again, due to the noise introduced in this database.

For case-based systems which have to deal with large case libraries, the high number of case comparisons may represent one of the biggest problems. Michie, Spiegelhalter & Taylor (1994b), who compared the performance of several classification systems (including decision trees, neural networks, the nearest-neighbour algorithm and others), concluded that the nearest-neighbour algorithm was the slowest of all methods tested, when dealing with large datasets. Our mechanism of retrieving cases by using the remindings provided by the neural network represents a new approach in reducing the number of case comparisons in CBR systems, which may also apply to making the nearest-neighbour algorithm faster when dealing with large datasets. Table 6.9 shows the reduction in case comparisons obtained in our experiments, which reached up to 91.4% for the soybean dataset.

This approach also represents an alternative form of narrowing the search space in terms of possible outcomes for a problem in CBR. When a new problem is being solved, instead of looking for a best match amongst all cases in the library (ie. cases for all the possible diagnoses), we try to make hypotheses (using the neural network), and search for similar cases within this set of hypotheses. This approach may lead the system to find a best-match which in fact is not the most similar case existing in the library. For instance the best possible match for a new AVSD case, according to the nearest-neighbour algorithm, is an ASD case previously diagnosed. However, as the remindings provided by the neural network do not indicate the ASD hypothesis, no ASD case will be looked for, and another previous AVSD case will be presented as the best-match. As demonstrated in the example, by narrowing the search space the system does not allow the nearest-neighbour method to look at cases individually without taking into account more general knowledge. This characteristic also enables the NN-CBR system to achieve a better performance when compared with the nearest-neighbour algorithm for the tests reported earlier in this chapter (tables 6.1 and 6.3).

Chapter 7

Discussion

This chapter is divided into two sections. The first section contrasts our CBR - neural network approach with other related work. Additionally, it comments further on the results obtained in the experiments presented in the validation chapter (chapter 6). The second section has the main goal of indicating the limitations of our hybrid approach, as well as identifying the types of applications where the option for a hybrid architecture may be suitable.

7.1 The NN-CBR Architecture and other Related Work

This section starts by analysing the main features of the NN-CBR architecture according to the same framework used to describe the hybrid CBR systems presented in chapter 2 (table 7.1).

	Level of integration	**Type of general knowledge**	**Priority in the reasoning process**	**Form of support**	**Knowledge embedded in cases and in generalisations**
NN-CBR	interactive back-up	compiled	neural network	neural network supports case retrieval	equivalent

Table 7.1: The main features of NN-CBR

NN-CBR operates with two different levels of integration: *interactive* and *back-up*. When operating with an *interactive* level, the neural network makes hypotheses and provides remindings for the CBR module in the selection of a previous case that can support one of the hypotheses. For problems where the neural network cannot make any hypothesis, the CBR module tries to take over and solve the problem on its own (*back-up* approach). This approach enables the system to deal with cases that would not be solved if an exclusively neural network approach were used. The reasoning process implemented gives *priority to the neural network.* As in Surma & Vanhoof (1995), this reasoning scheme is based on the assumption that we do not reason from prior cases when well-established rules are available (Riesbeck & Schank 1989). However, this reasoning mechanism does not present a high level of accuracy when diagnosing the more atypical cases for which the neural network cannot make hypotheses. The use of this method may be advantageous in applications where giving an answer is better than not reaching any conclusion. For example, in credit authorisation tasks where each application for credit has to receive some response. In this domain, the system would ideally produce an answer for all the credit demands, even if a certain priority would be given to one of the outcomes in order to reduce the risk of answers that could lead to loss of money (eg. giving priority to 'refuse credit authorisation'). For problems where the number of misclassifications should be kept as low as possible, this module could be deactivated to guarantee a lower number of errors.

Although having presented good results in the experiments described in chapter 6, giving priority to the general knowledge of the neural network may be problematic in some other circumstances. For instance, it may be not appropriate in domains where cases play a primary role, as in the domain of the English Common Law. Our target, however, has been to solve diagnostic problems in domains where general knowledge has a fundamental importance, and where facts can be transformed into rules when they are observed with a certain frequency.

The type of general knowledge kept in our architecture is *compiled,* and it is used for several purposes, from solving new problems to building explanations. This knowledge results from the learning process of the neural network and is stored in the network itself, as well as in the *diagnosis descriptors.*

Regarding the kind of support offered by the neural network, it provides the case-based component with remindings that are used to guide the search for similar cases. Therefore, the NN-CBR architecture uses different sources of knowledge in the reasoning process, most notably the generalised knowledge of the neural network and the specific knowledge stored in cases. This does not represent a novelty in itself, given that CBR systems have always employed other sources of knowledge in reasoning, be it generalisations of specific cases, or simply other types of information that can assist a system to solve the problems that it addresses. The system CHEF (Hammond 1986), for example, when creating a new recipe, determines the relationship between words using a dictionary that keeps a series of terms related to food, recipes and cooking. The system PERSUADER (Sycara 1988), whose goal is to generate compromise solutions in labour negotiations, uses inferential rules and heuristics to create adaptation tactics for solutions found in old cases. PROTOS (Bareiss 1989; Porter , Bareiss & Holte 1990), developed in the domain of clinical audiology, keeps a representation of a partial domain theory to assist it in determining category membership. However, there is a fundamental difference between the approach presented here and previous ones, which is concentrated on the way general knowledge is learned, represented, and (consequently) used.

Many early CBR systems, such as CYRUS (Kolodner 1983a), CASEY (Koton 1988) and MEDIC (Turner 1989), have compiled experiences in generalised episodes. These generalisations have been used to organise the case library and index the cases by exploiting the differences among them[1]. In NN-CBR, general knowledge learned from cases is represented in both *diagnosis descriptors* and neural networks, and used to draw hypotheses for possible solutions, to index the cases in the library and to explain the reasoning performed by the system. The cases in the library, representing specific knowledge, are used to support a particular hypothesis (or to choose among suggested hypotheses) and to present to the user previous cases that resemble the current one, highlighting their similarities and implying that

[1]The theory of *Dynamic Memory* postulates that a memory contains structures that keep general knowledge and that arrange (in a complex hierarchy) specific instances of the general knowledge. These memory structures are called Memory Organisation Packets (MOPs) and have been used in most of the early CBR systems. See Riesbeck & Schank (1989) for a more detailed explanation of MOPs and a description of a number of systems implemented using these memory structures.

the solutions outlined for those cases should also apply for the new one.

The NN-CBR inference and indexing schemes can be compared to those of PROTOS. To select and order the exemplars to match with a given case, PROTOS uses three types of indexing knowledge: remindings, prototypicality and exemplar differences. A reminding is a cue to a case classification. A reminding from a feature to a category suggests that the category is the most general classification for cases described with the feature. A reminding from a feature to an exemplar suggests that the exemplar will match cases described with the feature. The second type of indexing knowledge, prototypicality, orders the exemplars within a category according to their success in previous classifications. The third type of indexing knowledge, exemplar differences, indexes exemplars by the features that distinguish them from exemplars with similar descriptions.

When searching for an exemplar that matches a new case, PROTOS first collects remindings to categories. The most highly ranked categories are selected, and then the most prototypical exemplars for each category are retrieved. There is a basic difference in the PROTOS and NN-CBR inference processes. NN-CBR also starts by selecting the categories (making hypotheses), but then, NN-CBR does not look for the most prototypical cases within those categories. Instead, it considers any case in the library containing a set of findings detected to be important by the neural network. The assumption here is that even atypical cases may well provide the best match and the correct categorisation for a new case, and possibly also the most accurate explanation. This can be supported by the premise that one of the types of cases we store in our memories is *stories* (Riesbeck & Schank 1989), i.e. cases that have significance because of the extent of their difference from other cases. These cases tend to stand alone in the memory and represent the key to making predictions about special circumstances.

NN-CBR also uses a different approach for learning its indexes. While PROTOS index learning is based on the analysis of experts' given explanations, NN-CBR's is based on the calculation of how important each finding is in the categorisation process.

When compared with other hybrid approaches combining CBR with neural networks, the main peculiarity of NN-CBR is that its neural and its case-based components are somewhat independent. The majority of the approaches to combining CBR with neural networks integrate the two reasoning mechanisms by keeping cases as an intrinsic part of the neural network. This frequent choice of architecture is due to the fact that it can provide good solutions for CBR indexing and retrieval problems. For instance, cases can be represented in intermediate nodes of the neural network, being indexed by the network learning mechanism, and retrieved by the network consultation mechanism. Differently from these architectures, NN-CBR does not keep cases as part of the neural network. Instead, two separate NN and CBR modules work independently and the reasoning process is interleaved between the two modules. Furthermore, our hybrid approach deals with other problems faced by neural networks that are not treated in previous published research on the combination of CBR and neural networks. For instance, the interpretation of the mathematical knowledge of the neural networks into symbolic *diagnosis descriptors*, and the construction of explanations of reasoning using these descriptors. The main common point between our architecture and the architectures presented in chapter 2 is that our indexing scheme to select cases in the library is also based on the use of the neural network.

Other more traditional case-based indexing methods, such as explanation-based indexing (Kolodner 1993), analyse cases individually in order to identify factors that may lead to successful or unsuccessful situations, and try to use these factors as indexes. Our approach is once again different in that it relies more on the computation of recurrent or unusual situations, which is a task assigned to the CNM network. The indexes identified by this computation are then used to guide the search for similar cases in the library. The results obtained in our experiments, and presented in chapter 6, demonstrate that two indexing procedures have been highly efficient in retrieving relevant previous experiences. The first scheme uses remindings provided by the neural network to retrieve previous cases that can support a single diagnostic hypothesis. The second scheme uses remindings also provided by the neural network to retrieve similar cases that can be used to discriminate between competing hypotheses. The positive results presented in table 6.5 and 6.7 show that our indexing mechanism is capable of determining appropriate remindings in a completely automated way, and therefore may serve as an alternative indexing mechanism for case-based systems where any interaction with an expert for choosing remindings is difficult or costly.

Another research area that has some relation to the combination of neural networks with

CBR is the combination of neural networks with other symbolic reasoning mechanisms. The major reason for this other type of hybrid architecture is that in certain circumstances one single approach cannot provide all the necessary resources to represent knowledge and to reason in a given domain. Another important reason for combining symbolic and connectionist approaches is that there may be some cognitive plausibility in hybrid models. The CONSYDERR architecture (Sun 1995b), for instance, uses a two-layered knowledge representation to propose a reasoning model that mimics the biological neural interaction used to learn and reason with complex concepts composed of smaller units (microfeatures). Although it may be desirable to have models in which the components are as simple and homogeneous as possible, from an engineering perspective having a connectionist model with a hybrid configuration may be the only solution for a given application (Lange 1992).

Here, we have not been too concerned with the cognitive appeal of our architecture. Instead, we have concentrated our efforts on the development a hybrid structure that, although not serving as a theory for human cognition, can take advantage of various technologies to provide users with good knowledge representation mechanisms, learning capabilities and explanation facilities. Additionally, we wanted to respond to a well-known problem of neural networks, ie. the problem of representing in a symbolic fashion the numerical knowledge stored in the connections of the neural network. The system CONSYDERR provides a solution for this problem by connecting the nodes of its connectionist microfeature layer to the nodes of a symbolic concept layer. NEXTOOL also tackles this problem by connecting the nodes of the CNM network to *influence links* of a semantic network. However, neither NEXTOOL nor CONSYDERR enables the importance of findings (or feature/value pairs) to be represented in a graded symbolic scale. NN-CBR represents the importance of such findings in the ranking scheme provided by the *diagnosis descriptors*: *triggers* reference highly relevant findings, *primary findings* reference findings that are also important, and that can be frequently observed with the *triggers*, and *secondary findings* represent additional evidence which can be used to reinforce a diagnostic hypothesis.

Yang & Bhargava (1990) employ a scheme for setting the weights of a neural network which is approximate to our method representing knowledge in *diagnosis descriptors*. Their network has decision nodes that model certain types of correlation between attributes, namely: logical concurrence, negative concurrence, disjunction, exclusive disjunction, sufficient implication and necessary implication (see chapter 3 for a more detailed explanation). These decision nodes in fact cover a wider range of implications than our three *diagnosis descriptors'* attributes (*trigger*, *primary* and *secondary findings*), as they consider relationships of exclusiveness and negation between attributes. However, most of Yang & Bhargava's decision nodes have an 'all-or-nothing' character, which is scarcely observed in real world problems. Our attributes represent fuzzier relationships between findings, corresponding to several possible combinations of evidence that can be observed in the real world and that are captured and represented in the neural network.

Different classification schemes have been created to describe hybrid approaches combining connectionist and symbolic processes, as described in chapter 3. According to those schemes, our model could be classified as:

- tightly-coupled;
- intercommunicating;
- combinining a separate neural network and symbolic modules.

The main peculiarity of our approach when compared to other approaches to combining symbolic and connectionist processing is that we use a symbolic system (a frame hierarchy) with the main purpose of representing declarative knowledge about the domain, as well as some procedural knowledge which has a correspondence with the knowledge stored in the neural network. In this regard, our model is similar to NEXTOOL (Machado & Rocha 1992), which combined the CNM with semantic networks.

Our representation of the domain knowledge followed the representation used in HYCONES (Leao & Reategui 1993). However, it is also analogous to the representation of the domain knowledge used in other medical projects, such as POSCH (Long et al 1991). In this system, the medical and statistical knowledge stored consists of frames of observed data and calculated summaries. A frame characterises each variable (e.g. cholesterol), which is part-of a more complex concept (lipid profile, in this instance), which may have component parts (for example HDL, LDL and VLDL), a range of normal values, the relative importance, etc. We use the same abstraction concepts to describe the declarative knowledge of the application (ie. aggregation and generalisation). However, in our knowledge representation scheme we have also defined the *diagnosis descriptors*, which enable the frame hierarchy to represent both the declarative aspects of the domain knowledge, and the classificational knowledge used to solve the problems addressed.

Regarding the applicability of our approach, it has been shown in the validation step that NN-CBR is appropriate for some types of diagnostic and classification problems, but not for others (such as the classification problem of the second MONK's database). The NN-CBR model of reasoning is particularly suitable for domains such as medicine, where a diagnosis can be determined by the observation of some evidence, but never by the absence of evidence. For instance, in medicine, cancer cannot be diagnosed on the basis of the simple fact that a patient does not have *Down's syndrome* or *increased pulmonary flow* (symptoms of CHD). For this same reason the absence of findings is not considered in the construction of the knowledge graphs, on which the CNM and NN-CBR have been based.

It has also been observable from this experiment that when well-defined rules exist to solve a problem, it is not necessary to go to previous cases to confirm the hypothesis. During the testing of NN-CBR, for all the cases where the system considered only one diagnosis as a possible answer with a high degree of confidence, the search for a previous case to confirm the hypothesis was almost never necessary. However, the system would still look for a most similar case as a way of reassuring the user about the correctness of the final result and as a means of building a more sound explanation of the reasoning process.

The main contributions of this work have been the definition of the methods to build the *diagnosis descriptors* and the specification of how this type of general knowledge can be combined with CBR. This combination can be used to solve some of the problems of indexing and retrieval that are still current in CBR. Moreover, a similarity function computing the *relative goodness* of attributes has been defined and applied successfully, which offers a solution for case-matching problems where some findings are more important and discriminating than others. The *relative goodness* concept has been used originally as a measurement of how discriminating attribute/value pairs are in classifying datasets where data are inconclusive, ie. where the attributes used in describing a set of examples are not sufficient to specify exactly one outcome (class) for each example. As it has been demonstrated in some previous experiments, this technique has been able to achieve better results than other techniques such as ID3, or other statistical methods (Uthurusamy, Fayyad & Spangler 1991). In our experiments, the performance of the nearest-neighbour algorithm using the *relative goodness* measurement has achieved a very close performance to that of the decision trees (the latter using information theory to determine how discriminating an attribute is). Using the *relative goodness* has usually provided us with better results, though, which has made us opt for this measurement in our similarity function. However, a yet different similarity function may provide a better performance in another domain. In which case, this function can simply replace the similarity function which uses the *relative goodness*, without requiring further modifications in the hybrid architecture.

Another relevant possibility offered by our NN-CBR approach is that of combining the knowledge of an expert with knowledge extracted from cases. For example, in Leao & Reategui (1993) the knowledge graphs of experts and non-experts have been converted into CNM networks, and then refined, through the training of the networks with a given set of cases. In those experiments, the performance of the refined systems was significantly better than the performance of the neural networks created from the graphs and with no refinement (improvement from 42.5% accuracy to 78.7% in the case of expert's graphs). In our hybrid approach, combining knowledge graphs with the knowledge stored in cases is also possible: the graphs elicited from experts (or non-experts) can be converted into neural networks, which are trained with a given set of cases. These cases are stored in the case library and, subsequently, the neural networks are mapped into the *diagnosis descriptors*.

This possibility enables the construction of richer knowledge bases, as not only the knowledge of multiple experts can be taken into account during the collection of the knowledge graphs, but this knowledge can also be refined by using concrete experiences. These experiences are relevant in that they may report subtleties that are usually not mentioned by the experts when they try to explain the way in which they solve a given problem. The knowledge of experts can also be combined with machine-processed knowledge in the inverse way, ie. the information initially extracted from data can be reviewed and refined later by experts (McLeish et al. 1991). The main advantage of this approach is that the elicitation of knowledge from an expert is eased by having an automated first step of knowledge acquisition. Still on the combination of human and machine processed knowledge, machine learning and classification techniques may be able to make a substantial contribution to organising human knowledge, or even to manufacture new knowledge (Michie, Spiegelhalter & Taylor 1994b). For example, the system KARDIO's interpretations on ECG examinations do not contain a single rule of human authorship (Bratko, Mozetic & Lavrac 1989).

It is also important to stress here that the actual collection of cases is a crucial step in the construction of a case library that is to be used by an automated method to generate classificatory knowledge. As stated by Janet Kolodner (1993), a case-based system can only be as good as the quality of the cases it uses to solve new problems. In our experiments this claim was supported by the fact that the cases collected with the supervision of an expert led to a system that achieved a high level of accuracy, whereas the cases collected by a less experienced cardiologist were unable to help the resulting system to reach results as good as the former one. Moreover, its symbolic knowledge representations (*diagnosis descriptors*) were not so clear.

7.2 Limitations and Applicability of the Hybrid Approach

We have presented here a hybrid approach which combines case-based reasoning with neural networks in order to solve classification[1] problems. The classification tasks that may be learned by this approach are those in which the classes are known previously, and where the main goal is to establish how a new case can be classified into one of the existing classes. In this thesis, several examples of applications have been given to illustrate the use of the hybrid approach in solving diagnostic and classification problems, in domains as varied as congenital heart diseases, mushroom classification, soybean disease diagnosis and

[1]We consider diagnosis here as a type of classification problem, as the types of diagnoses we deal with always have a limited number of possible outcomes. For problems where the number of outcomes cannot be enumerated easily, diagnostic problems can be seen as a type of explanation task where different interpretations for a set of symptoms observed have to be given. For example, in psychiatric diagnosis, different causes may have to be identified to explain the psychological problems presented by a certain patient, as in SHRINK (Kolodner 1993).

classification of objects with complex sets of properties (the MONK's Problems). In all these applications, the main task at hand is the identification of a class (or diagnosis) according to a particular case description.

The hybrid architecture NN-CBR introduced a number of facilities related to knowledge acquisition and refinement, representation of declarative and heuristic knowledge, and explanation of reasoning. However, there is a certain cost for the provision of such facilities, which is related to the fact that independent reasoning components (neural network and CBR) have to be coordinated. The problems related to having more than one component involved in the reasoning process are the following:

- *redundancy of information*: as the *diagnosis descriptors* are used to represent the knowledge stored in the neural networks, both structures keep the same type of classification knowledge. Moreover, the cases stored in the case library may be seen as one more different form of representation of the same type of knowledge. This redundancy of information leads to two major problems:

  1) more storage space is needed to keep all these memory structures, namely the neural network, the case library and the *diagnosis descriptors*. In our software prototype, these structures are kept in the live memory, which may represent a problem for applications dealing with large case libraries. The use of a database management system (DBMS) to store the case library could minimise this problem, though. The need for integrating DBMSs within CBR methodologies has already been discussed by Brown, Watson & Filer (1995), where the authors show that, for practical reasons, it is essential that CBR tools can access and use data which are kept in existing databases. In such applications, cases must no longer be considered as concretely represented at the data level, but as virtual views of the underlying data. We believe that the possibility of accessing data stored in commercial database systems would not only make the use of the NN-CBR architecture less subject to the sizes of case libraries, but it would also enable the quicker development of real world applications for which a database storing previous experiences already exists. This is a whole new research area that is outside the

intended scope of this thesis.

2) as for the neural network, *diagnosis descriptors* and cases stored in the library keep correlative knowedge; when one of them is modified, all the others have to be updated. This course of action is necessary to avoid the problem of having self-contradictory knowledge stored in the different knowledge sources. At the implementation level, procedures (such as daemons) could be used to update all knowledge sources when one of them is modified. For instance, when a new case is inserted in the library, a procedure could be activated to refine the neural network according to the case, and to recalculate the *diagnosis descriptors*, in order to harmonise the knowledge of all three structures. However, these computations may have an undesirable overhead cost. Our recommendation for attenuating this cost is that the case library would not be incremented at the occurrence of every new case. Instead, it would only be modified after the observation of a certain number of cases. This would compel the neural-network learning mechanism, as well as the recomputation of the *diagnosis descriptors,* to be activated only at certain times, which would reduce the time necessary to keep the knowledge of the three structures harmonised. Furthermore, having control of when to activate the learning procedures would enable the user to keep a firmer hand on the performance of the system. For example, in situations where a satisfactory performance is regularly obtained, the system's learning mechanism should not be activated, as this could represent a change in behaviour and a possible drop in the performance.

• *cognitive plausibility:* it is more difficult to design a plausible problem-solving mechanism in an architecture that uses different knowledge representation and reasoning techniques. As observed by Lange (1992), it is desirable for an architecture to have components as simple and homogeneous as possible. However, from an engineering perspective, having a hybrid scheme combining different representation and reasoning mechanisms may well be a good solution for a given application (or certain types of applications). We have concentrated our efforts on this goal, ie. benefiting from the advantages that could be provided by each of the technologies used, instead of trying to build a well-founded model of human reasoning. However, from a more abstract point of view, we have also based our work on plausible claims. Firstly, we have tried to use general as well as specific knowledge in our reasoning mechanism. From a cognitive point of view, using both general and specific knowledge is more plausible than relying solely in some type of compiled knowledge. There is no shortage of confirmations from psychology (Estes 1984, Sternberg 1984) that people keep in their memories experiences represented in single episodes, as well as summary representations of these experiences. Both types of knowledge are used to understand and solve a newly-posed problem here. Another example of the cognitive plausibility found in our reasoning mechanism is the priority given to the general knowledge kept in the neural networks. According to Riesbeck & Schank (1989), we only reason from prior experiences when well-established rules are not available. We have done something analogous in using the neural network to determine the solution to a given problem when it indicates only one hypothesis with a high level of confidence. Previous experiences are called up to solve a problem on their own only when the neural network cannot give any clue on which hypotheses to consider.

In addition to the difficulties reported, in domains where cases play a primary role (such as in law), giving priority to the generalised knowledge of the neural networks may not be the best policy. Although this priority could be modified in NN-CBR to have the reasoning process start with the case-based procedure, the inversion of priority levels would not make much sense in our approach. The NN-CBR architecture is grounded on the idea that the generalised knowledge of the neural network can guide the CBR system in finding a best match, and give the reasoning process a more global view of the problem's solution space. As observed by Barletta (1994), by looking at previous cases individually when solving a new problem, the nearest-neighbour algorithm may find a most similar previous case that is only a local best match, but which is not optimal. Our reasoning approach forces the system to take into account the similarity of a given new case with all the possible classes first, and only at a second moment to retrieve cases for comparison. The advantage of having such a reasoning method is that the nearest-neighbour algorithm used by the CBR component becomes less susceptible to errors when a case library containing noisy cases is analysed. This advantage would no longer exist if the priority levels were inverted to benefit the case-based approach.

The NN-CBR architecture is also not the most appropriate for applications that have to deal with adaptation problems, ie. the modification of a solution obtained from a previous case in order to *reuse* it in a new case. This *reuse* step is often needed in design and planning tasks, where a complex previous solution has to be modified according to requirements and constraints of the new case. As we have not targeted such types of applications, no adaptation facilities are provided in our architecture. However, adaptation procedures could be added to NN-CBR. This would enable the resulting system not only to diagnose a case, but also to propose courses of action to solve the problems diagnosed. For instance, in the CHD problem, additionally to identifying CHD diseases, the system could propose treatments based on successful previous experiences. Another example could be in fault-diagnosis applications, where the system would be able to identify the problem causing the malfunctioning of a certain device, and propose a course of action to solve the problem.

When amending adaptation procedures with the NN-CBR architecture in mind, the structure of the cases would have to be changed. They would have to include, for instance, course of actions and outcomes, showing successful and unsuccessful solutions. The main difficulty in amending adaptation procedures is that our measurements for finding a best-matching case do not consider that a solution would have to be changed to fit a new case. The adaptation requirements of candidate cases during retrieval should also be taken into account in some applications, as reported in Déjà Vu when regulating the action of automomous vehicles within real industrial environments (Smyth & Keane 1995). As has been demonstrated by Smyth and Keane, the use of adaptation-guided retrieval can reduce significantly the adaptation costs, as the most adaptable (instead of the most similar) cases are selected and preliminary adaptation work is performed during case retrieval. It would be difficult to take into account such adaptation requirements within our model of reasoning. Thus, for applications where the adaptation costs are too high, our architecture should not be considered.

7.2.1 Modifications and Possibilities

Despite the problems reported, the use of different components in representing knowledge and in reasoning may bring some interesting benefits, other than the advantages provided separately by each of the techniques used. For instance, implemented systems can be combined in a hybrid architecture, each of them being adjusted individually to produce an optimal performance[1]. And in situations where one of the components gives an unsatisfactory performance, it may be replaced by a yet different component, without causing too many changes in the rest of the architecture. A practical example in our architecture can be given by our CBR component. Its similarity function is based on the *relative goodness* concept. In applications where this measurement of the discriminatory capacity of attribute/value pairs does not produce good results, the CBR component can be replaced by another case-based mechanism using a different similarity function (e.g. one using information theory, as in ID3). In NN-CBR, the replacement of the CBR component does not demand further changes in the architecture.

Replacing the neural network, however, would cause much deeper modifications, as the *diagnosis descriptors* of the symbolic components have been created specifically to represent the knowledge of the CNM network. An example of a hybrid architecture that enables the use of different neural networks is HYCONES II (Leao, Reategui, Guazzelli & Mendonca 1994). This system incorporates the model SMART (Guazzelli & Leao 1994) in its structure, and enables the user to choose whether he wants to use SMART or CNM to solve new problems. However, in HYCONES II there is no analysis of the neural networks in order to build symbolic descriptors. The analysis of the CNM in NN-CBR is what makes the replacement of the neural network by other connectionist models impracticable (if no serious changes in the rest of the architecture are wanted).

Another replacement one can think of in the NN-CBR arcuitecture is the substitution of the frame system by another mechanism for knowledge representation. For instance, the frame hierarchy used to represent the domain knowledge could be replaced by a semantic network, as in NEXTOOL (Machado & Rocha 1992). Although the abstraction concepts used in our

[1]This is not valid for hybrid architectures of the *fully-integrated* type.

frame mechanism provide a significant amount of flexibility for representing the domain knowledge, the user may be more familiar and feel more comfortable with relying on semantic networks. The replacement of the frame system in NN-CBR could be achieved through the mapping of one knowledge-representation mechanism into another. For instance, the abstraction concepts of the frame hierarchy could be mapped into *is-a* (generalisation and classification), *part-of* (aggregation) and *member-of* (association) arcs of the semantic network. The *diagnosis descriptors*, which could be more difficult to substitute, could be represented in the *influence links* of NEXTOOL's semantic network. However, different types of *influence links* would have to be used in order to represent the *diagnosis descriptor's triggers*, *primary* and *secondary findings*.

We consider these to be interesting possibilities that could customise our architecture to particular wishes and requirements of different users. The next subsections point out some problems that have to be watched out for when developing a system with our hybrid approach.

7.2.2 Creating the Neural Network

One of the problems one may encounter when building the CNM network is that too many combinatorial nodes may be generated to represent the influence of the different combinations of evidence for the diagnoses (or classes) considered in the application. This kind of adverse effect should be expected especially in applications where there is a large number of items of evidence, as well as diagnoses (or classes).

To avoid this problem, the user should not start the construction of the neural network with a fully-connected network (ie. the user should not try to combine all the evidence considered, in all the possible ways, for each of the diagnoses). Instead, only the findings observed for a particular diagnosis should be used to build the clusters that indicate the same diagnosis. This method should reduce significantly the number of combinatorial nodes created, as well as the time to construct the complete neural network. Alternatively, a different mechanism that produces only the relevant clusters of evidence can be used, as in Machado, Rocha & Denis (1992), where a genetic algorithm generates and selects the most significant clusters for the identification of each diagnosis.

Another difficulty that one may have to face in the use of the CNM is in the adjustment of the *Tacc* parameter. Originally, this parameter is intended to set a threshold for the acceptance of answers coming from the neural network, as well as to define the minimum value for pathognomonic connections (ie. those that have only rewards but no punishments) at the time of the construction of the neural network. In NN-CBR, *Tacc* is only used to build the neural network, as we exploit all the answers coming from the network to build remindings, whether they have a high or low confidence degree. The value we have chosen for *Tacc* has been 0.5, which means that pathognomonic connections with a very small number of rewards will not have their weights set with values lower than 0.5. In applications where the user knows the classifications of previous cases to be always correct, and where he believes pathognomonic pathways can be assigned a more important role, the value of *Tacc* may be increased. It is important to stress here that this increment would make the neural network more susceptible to noise, though. A high value assigned to *Tacc* would produce a strong weight in a connection with only a few rewards (and we should consider the possibility that these few rewards may in fact come from cases classified badly). The extreme situation is where *Tacc* is assigned the value 1, which would make any connection of the network containing no punishments to also carry the highest possible weight, therefore making pathognomonic pathways into determining factors in the diagnostic process. A smaller value of *Tacc* improves the ability of the system to cope with noisy datasets.

7.2.3 Building the Diagnosis Descriptors

Selecting the *trigger*, the *primary findings* and the *secondary findings* of the *diagnosis descriptors* is an important step in the construction of a system with the NN-CBR architecture. These attributes are selected according to the findings that appear in nodes carrying a high connection weight or a high number of rewards in the neural network. It is not difficult to determine the *trigger* for a diagnosis, as it appears in the node of the network possessing the highest connection weight for the same diagnosis. It may, however, be more difficult to define *primary* and *secondary findings*. We have used a threshold mechanism to determine the *secondary findings* of a diagnosis. This mechanism chooses as *secondary findings* the evidence appearing in nodes of the neural network with connection weights bigger than a given threshold. The value chosen for this threshold has been the value of *Tacc*, as *Tacc* determines the connections of the neural network able to produce useful answers.

We have also used a threshold mechanism to determine the *primary findings* of a diagnosis, by selecting as *primary findings* the evidence represented in nodes that appear in at least half of the training examples. This is justified by the fact that the findings observed in most of the cases for a particular diagnosis are generally expected to be present in other cases with that same diagnosis, whether they are specific to the diagnosis or not. This method was able to build the *diagnosis descriptors* for the first three diagnoses appropriately (ASD, AVSD and VSD). For the other 4 diagnoses, no findings (or combinations of findings) could be observed with a frequency higher than the one requested (50%). In such cases, we opted for informing the user of the problem, and for continuing the construction of the *primary findings* through the selection of the four most frequent findings (located in nodes of the network with the highest reward accumulators). We have chosen the number 4 as the maximum number of *primary findings* because, according to the knowledge graphs elicited from experts and non-experts, clusters with more than five elements are scarcely seen in the leftmost trees of the graphs (the trigger references one of these elements, while the *primary findings* would reference up to four other elements). Additionally, five is the number that has been suggested by Machado & Rocha (1989) as the maximum number of items of evidence to be clustered in each combinatorial node of the CNM, making reference to the magical number 7 plus or minus 2 (Miller 1956).

The use of this alternative method for the selection of the *primary findings* also serves as a warning for the user when the system cannot find enough evidence to operate its ordinary procedure for the selection of *primary findings*. What has to be considered, though, is that by not enforcing a minimum level of frequency in the alternative selection of the *primary findings*, the risk of selecting irrelevant evidence as *primary findings* is increased. Thus, in such situations the user should always verify the correctness of the *diagnosis descriptors*, and try to rectify possible errors by adjusting the threshold that enforces the minimum level of frequency required for *primary findings*.

Chapter 8

Conclusions

We have presented an architecture that contains a new way of integrating CBR and neural networks. While a neural network is used to make hypotheses and to guide the search for similar cases in the library, CBR is used to select a most similar match for a given problem, supporting a particular hypothesis or deciding among hypotheses. Items called *diagnosis descriptors* are created and maintained according to the knowledge stored in the neural network, keeping an intelligible description of the knowledge represented in the network and defining a ranking scheme for the most important attributes observed in these cases (*trigger, primary findings* and *secondary findings*). This ranking scheme is used for consultation purposes, for confirming or refuting a final result, and for building explanations.

A system for the diagnosis of Congenital Heart Diseases (CHD) has been presented and evaluated. Together with other tests using the *mushroom database*, the *soybean database* and the *MONK's Problems* database, it has been possible to conclude that:

- the performance of the hybrid NN-CBR system is always equal or superior to the performance of the CNM network. From tables 6.1, 6.3, 6.6 and 6.8 of the previous chapter, we can see that the hybrid system has been able to introduce an improvement in the CNM system's performance of up to 14.9% (in the number of correct answers).

- the indexing scheme presented here, based on remindings provided by the neural network, is promising for guiding the search for the most relevant cases in the library when a best match is demanded. Tables 6.5 and 6.7 show the performance of the indexing scheme in solving problems related to four different databases. The percentage of 93.3 correct answers has been reached for the substantial test database

Chd3, 76.12% for database *Chd7*, 98.89% for the *mushroom classification problem*, and 93.59% for the *soybean disease* database.

- the indexing scheme based on the remindings provided by the neural network is also promising for narrowing the search space and reducing the number of case comparisons when a best match is demanded. Table 6.9 shows the significant reduction in the number of case comparisons introduced by the hybrid NN-CBR approach, resulting in as much as 91.4% less comparisons in the *soybean* problem (ie. an average of 25 case comparisons was necessary to solve the problems at hand, instead of 290 if a standard nearest-neighbour procedure had been used).
- the knowledge learned and represented in the *diagnosis descriptors* is similar to that of experts when the cases used to teach the system have complete and correct descriptions. Figures 6.1 to 6.7 of the previous chapter show that the *diagnosis descriptors* built through the processing of the cases collected with expert supervision a had a very similar representation to that of the mean knowledge graphs elicited from multiple experts.
- the *diagnosis descriptors* can give the user a clearer idea about the knowledge stored in the neural network, and can be used to build explanations for the reasoning process carried out when finding suitable answers for new cases. As shown in section 5.8 of chapter 5, these explanations take into account the importance of the findings present in a problem case, according to their characterisation in the *diagnosis descriptors.*
- for some types of classification problems, the method presented may not be the most appropriate, as the neural network and the *diagnosis descriptors* cannot represent explicitly some logic operations between attributes that may be necessary in some classification tasks, as in the *MONK's Problem* number 2.

Along with other CBR systems, NN-CBR has the following advantages:

- it may be able to ease the task of knowledge acquisition, as most of the knowledge needed for solving new problems is contained in cases that have already been solved.
- it can learn incrementally through the incorporation of new cases in the library, the

training of the neural network with the same cases, and updating of the *diagnosis descriptors*.

- it has a rather plausible reasoning mechanism where both general and specific knowledge are used in the problem-solving process. General knowledge is represented in the neural network and *diagnosis descriptors,* and used to make hypotheses for newly-posed problems. Specific knowledge, in contrast, is represented in the cases stored in the library, and used to confirm a given hypothesis or to discriminate between hypotheses.

In addition to profiting from the advantages provided by the case-based approach, the NN-CBR arcuitecture is also able to minimise some significant difficulties generally faced by more traditional CBR architectures. By using the neural network to guide the search for similar previous cases, our hybrid system reduces considerably the number of case-comparisons when solving a new problem, therefore shrinking some of the difficulties of dealing with large case libraries. Furthermore, by using the remindings provided by the neural network to retrieve the most relevant experiences, the NN-CBR architecture does not allow the existence of a vast number of cases to bias it too much in the problem-solving process.

The main drawbacks of the NN-CBR approach are related to the fact that independent reasoning components (neural network and CBR) have to be coordinated, which can represent a difficult task. Additionally, when the knowledge of one component is modified, the others have to be updated. For instance, when new cases are added to the system's library, the neural network has to make computations on these cases and the *diagnosis descriptors* have to be recalculated, which can have a certain overhead cost.

When compared to other approaches to combining CBR with neural networks, our approach is different as it does not keep cases as an intrinsic part of the neural network. Instead, two separate NN and CBR modules work in an independent fashion. Furthermore, it deals with other problems faced by neural networks that are not treated in previous published research on the combination of CBR and neural networks. For instance, the interpretation of the mathematical knowledge of the neural networks into symbolic *diagnosis descriptors*, and the construction of explanations of reasoning using these descriptors. The main common point between our architecture and the architectures presented in chapter 2 is that our indexing scheme to select cases in the library is also based on the use of the neural network.

The main contributions of this work have been:

- the conception of the particular hybrid architecture combining CBR with the CNM. This hybrid architecture benefits from the knowledge-representation comprehensibility of symbolic systems, the learning ability of connectionist systems, and the explanation capability of symbolic and case-based systems.

- the specification of the reasoning process where an alternative indexing scheme that calls on the CNM network is used. The analysis of each part of the reasoning mechanism in NN-CBR can be valuable for other researchers working with alternative CBR indexing and retrieval schemes.

- the definition of the methods to interpret and map the knowledge of the neural network into *diagnosis descriptors*. This is a noteworthy contribution to analysis knowledge stored in neural networks, which is a current area of research in neural networks and hybrid connectionist systems.

- the construction of a system for the diagnosis of seven common CHD problems, as well as the development of other experiments that have proposed different ways for combining general and specific knowledge in various domains, namely: classification of credit card transactions (Reategui & Campbell 1995); identification of the profile of patients awaiting heart transplantation (Reategui, Campbell & Borghetti 1995); and diagnosis of three congenital heart diseases (Reategui, Campbell & Leao 1996), the latter having an indexing scheme which did not use neural networks, but simply the information contained in the *diagnosis descriptors.*

An interesting subject for further research is the investigation of the behaviour of NN-CBR when applied in domains with weak theories, where there are no defined rules and where the formation of the *diagnosis descriptors* could improve the process of solution for problems with limited amounts of theory-enforced structure. Using NN-CBR to solve other real-world problems could provide a more exhaustive view of the generality of our approach, and possibly also suggest insights for how the model could be enhanced from a cognitive point of view.

Another extension of NN-CBR could be the redefinition of the *diagnosis descriptors* at a higher level of detail, which would require the use of other statistical methods for the analysis of the neural network (e.g. analysis of the weight distribution of the network). Furthermore, a mechanism that would enable the system to learn *negative findings* through the identification of findings that commonly lead to misclassifications could be constructed. This idea has been based on PROTOS' method for learning censors, i.e. remindings that identify negative associations between case findings and exemplars. PROTOS learns its *censors* through the search for any mutual-exclusion relations used in explanations provided by an expert.

Regarding the reasoning process, an interesting topic for further investigation is the search for alternative reasoning steps when an answer is intercepted for lack of credibility. For instance, the system could use other indexing cues to find other previous cases that bear similarities to the problem case. At the moment, NN-CBR simply states that it is not possible to solve the problem. Another way of improving the performance of the system would be to investigate how to identify 'noisy cases' in the library, which would enable NN-CBR to avoid the problem of retrieving inappropriate cases due to previous misclassifications.

We believe that research in this direction can originate more powerful diagnostic and classification systems. The investigation of the interconnections of this work with cognitive science may also lead to better simulations of the reasoning processes observed in humans, and thus make some contribution to the development of general architectures for cognition.

Appendix A

The knowledge Acquisition Methodology for the Elicitation of Knowledge Graphs[1]

A1. Phase 0 - Problem Definition

1) Define the project scope (problem domain).
2) Define the diagnostic hypotheses that will be part of the project.
3) Define a list of symptoms, signs and test results, with the assistance of an expert (or a group of experts).

A2. Phase 1 - Knowledge Acquisition

1) Select one of the diagnostic hypothesis.
2) Ask the expert to indicate, in the list of symptoms, the items necessary for the formulation of the diagnostic hypothesis, by which a working subset of symptoms is defined.
3) Ask the expert to rank the working subset according to the importance of the items for the diagnostic.
4) Ask the expert to assume the items of the ordered list as the evidence nodes of a knowledge graph and to associate them in the form that he judges necessary to establish an adequate foundation for the diagnostic hypothesis (generating intermediate nodes that converge to the diagnostic hypothesis).

[1]From Machado, Rocha & Leao (1990).

5) Ask the expert to ascribe a membership degree between 0 and 10 for the information represented in each node of the graph in relation to the diagnostic hypothesis.

6) Ask the expert to define the logical operators (AND, OR, NOT) associated with the nodes of the graph.

7) Repeat the process for the other diagnostic hypotheses, to form a family of graphs.

Appendix B

The Learning Algorithms of the Combinatorial Neural Model[1]

B1. Neural Network Punishment and Reward Algorithm (scratch version)

Given the evidential flow of a neural network connection as the product:

- (activation of the origin node x connection weight), for an excitatory connection;
- ((1-activation of the origin node) x connection weight)) for an inhibitory connection;

For *importance degree* the relevance of an example in the learning process, Do:

```
• Set all the punishment and reward accumulators of the network to 0;
• Set all the weights of the network to 1;
• For each case of the training set Do:
  - Propagate the evidence from the input nodes to the output layer, according to the
    activation functions of the CNM;
  - For each pathway reaching a diagnosis Do:
    If the diagnosis reached corresponds to the correct diagnosis of the case
    Then
      navigate backwards from this node to the input nodes, increasing the reward
      accumulators of each traversed connection by the product:
      evidential flow x destination node activation x importance degree
```

[1] From Machado, Rocha & Denis (1992).

```
Else
    navigate backwards from the output node to the input nodes, increasing the
    punishment accumulators of each traversed connection by the product:
    evidential flow x destination node activation x importance degree
```

B2. Neural Network Punishment and Reward Algorithm (refinement version)

```
For each training example, Do:
    • Propagate the evidence from the input nodes to the output layer, according to the
      activation functions of the CNM;
    • For each hypothesis H that is a solution for the case, Do:
        - Calculate $\alpha = 1$ - activation of (H)
        - For each pathway reaching the hypothesis H, Do:
            navigate backwards from this node to the input nodes, increasing the reward
            accumulators of each traversed connection by the product:
            $\alpha$ x evidential flow x destination node activation x importance degree
    • For each hypothesis H that is not a solution for the case, but which has an activation
      higher than a pre-established threshold (Tacc), Do:
        - Calculate $\alpha$ = activation of (H)
        - For each pathway reaching the hypothesis H, Do:
            navigate backwards from the output node to the input nodes, increasing the
            punishment accumulators of each traversed connection by the product:
            $\alpha$ x evidential flow x destination node activation x importance degree
```

B3. Pruning and Normalization algorithm

```
For each connection in the network Do:
  • Compute the net accumulator value as:
        $netacc = reward\ accumulator - punishment\ accumulator$
  • If $netacc <= 0$ then remove the connection of the network
  • If the punishment and the reward accumulators have the same value, then set the connection weight to 0;
  • If punishment accumulator > 0
      Then
          compute the connection weight as $netacc / maxnet$, where $maxnet$ is the maximum netacc existing in the network for that diagnosis
      Else
          compute the connection weight as:
                  $sqrt(Tacc) + (1 - (sqrt(Tacc))) \times netacc / maxnet$
          where $Tacc$ is the threshold that determines the minimum output for an answer to be accepted
  • If connection weight < pruning threshold, delete the connection
Remove all nodes that lose access to diagnosis nodes.
```

Bibliography

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

Alpaydin. E. (1991). Gal: Networks that grow when they learn and shrink when they forget. Technical report TR 91-032, *International Computer Science Institute,* France.

Auriol, E., Manago, M., Althoff, K-D., Wess, S. & Dittrich, S. (1995). Integrating induction and case-based reasoning: methodological approach and first evaluations. In Haton, J-P., Keane, M. & Manago, M. (eds.) *Advances in Case-Based Reasoning: Second European Workshop EWCBR-94*, 18-32, Chantilly, France. Berlin, Germany: Springer Verlag.

Auriol, E., Wess, S., Manago, M., Althoff, K-D. & Traphoner, R. (1995). INRECA: A seamlessly integrated system based on inductive inference and case-based reasoning. In Veloso, M. & Aamodt, A. (eds.), *Case-Based Reasoning: Research and Development. First International Conference, ICCBR-95*, 371-380, Sesimbra, Portugal. Berlin, Germany: Springer Verlag.

Azcarraga, A. & Giacometti, A. (1991). A prototype-based incremental network model for classification tasks. *Fourth International Conference on Neural Networks and their Applications*, Nimes, France.

Bamberger, S. K. & Goos, K. (1993). Integration of case-based reasoning and inductive learning methods. In Wess, S., Althoff, K-D. & Richter, K. (eds.), *Proceedings of the First European Workshop on Case-Based Reasoning*, 296-300, Kaiserslautern, Germany.

Barletta, R. (1994). A Hybrid indexing and retrieval strategy for advisory CBR systems built with Remind. In Haton, J-P., Keane, M. & Manago, M. (eds.). *Proceedings of the Second European Workshop on Case-Based Reasoning*, 49-58, Chantilly, France. Paris, France: Acknosoft Press.

Bareiss, E. R. (1989). The experimental evaluation of a case-based learning apprentice, In Hammond, K. J. (ed.). *Proceedings of the Case-Based Reasoning Workshop*, 162-167, Pensacola Beach, FL. San Mateo, CA: Morgan Kaufmann.

Bareiss, R. (1989). *Exemplar-based knowledge acquisition.* San Diego, CA: Academic Press.

Bratko, I., Mozetic, I. & Lavrac, L. (1989). KARDIO: A study in deep and qualitative knowledge for expert systems. Cambridge, MA: MIT Press.

Becker, L. & Jazayeri, K. (1989). A connectionist approach to case-based reasoning, In Hammond, K. J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, 213-217, Pensacola Beach, FL. San Mateo, CA: Morgan Kaufmann.

Becraft, W. R., Lee, P. L. & Newell, R. B. (1991). Integration of neural networks and expert systems for process fault diagnosis. *Proceedings of the International Joint Conference on Artificial Intelligence*, 832-837. San Mateo, CA: Morgan Kaufmann.

Bookman, L. A. (1995). A Framework for integrating relational and associational knowledge for comprehension. In Sun, R. & Bookman, L. A. (eds.), *Computational Architectures Integrating Neural and Symbolic Processes: a Perspective on the State of the Art*, 283-317. Dordrecht, Netherlands: Kluwer Academic.

Branting, L. K. (1989). Integrating generalizations with exemplar-based reasoning, *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, 139-146. Northvale, NJ: Erlbaum.

Brown, M., Watson, I. & Filer, N. (1995). Separating the cases from the data: towards more flexible case-based reasoning. In Veloso, M. & Aamodt, A. (eds.), *Case-Based Reasoning: Research and Development. First International Conference, ICCBR-95*, 157-168, Sesimbra, Portugal. Berlin, Germany: Springer Verlag.

Carbonell, J. G., Michalsky, R. S. & Mitchell, T. M. (1986). An overview of machine learning. In Michalski, R. S., Carbonell, J. G. & Mitchell, T. M. (eds.), *Machine Learning: an Artificial Intelligence Approach*, vol 1. Palo Alto, CA: Tioga.

Carbonell, J. G. (1989). Introduction: paradigms for meahine learning. *Artificial Intelligence*, 40(1-3): 1-9.

Carpenter, G. A., Grossberg, S. & Iizuka, K. (1992). Comparative performance measures of Fuzzy Artmap, Learned vector quantization, and backpropagation for handwritten character recognition. *International Joint Conference on Neural Networks*, 1794-1799, Baltimore, MD.

Charniak, E. (1986). A neat theory of marker passing. *Proceedings of the AAAI*, 584-588, Phyladelphia, PA. Cambridge, MA: AAAI Press/ MIT Press.

Crevier, D. (1993). *AI: The Tumultuous History of the Search for Artificial Intelligence*, New York, NY: BasicBooks.

Denis, F. A. R. M. & Machado, R. J. (1991). *O Modelo Conexionist Evolutivo*, Technical Report CCR128, IBM Rio Scientific Center, Rio de Janeiro, RJ, Brazil.

Dinsmore, J. (1992). Thunder in the gap. In Dinsmore, J. (ed.), *The connectionist and the symbolic paradigm: closing the gap*, 1-23. Hillsdale, NJ: Erlbaum.

Estes, W. K. (1984). Learning, memory and intelligence. In Sternberg, R. J. (ed.), *The Handbook of Human Intelligence*, 170-224. Cambridge, England: Cambridge University Press.

Feng, C. & Michie, D. (1994). Machine learning of rules and trees. In Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (eds.), *Machine learning, neural and statistical classification*, 50-83. Chichester, England: Ellis Horwood.

Feret, M. P. & Glasgow, J. I. (1993). Hybrid case-based reasoning for the diagnosis of complex devices. *Proceedings of the AAAI*, 168-175, Washington, DC. Menko Park, CA: AAAI Press.

Fikes, R. & Kehler, T. (1985). The role of frame-based reasoning. *Communications of the*

ACM, 28(9): 904-920.

Fodor, J. & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: a critical analysis. *Cognition*, 28: 3-71.

Friedman, J. H., Bentley, J. L. & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3: 209-226.

Gallant, S. I. (1988). Connectionist Expert Systems. *Communications of the ACM*, 31(2): 152-169.

Georgin, E., McDonald, J., Ricard, B. & Jacquot, J-P. (1994). The use of cases in diagnostic explanations. *Proceedings IEE Colloquium on Case-based reasoning: prospects for applications* - digest 055, pages 2\1-2\3.

Goonatilake, S. & Khebbal, S. (1995). Intelligent hybrid systems: issues, classifications and future directions. In Goonatilake, S. & Khebbal, S. (eds.). *Intelligent Hybrid Systems*, 3-20. New York, NY: John Wiley.

Greco, G. & Rocha, A. F. (1987). The fuzzy logic of text understanding. *Fuzzy Sets and Systems*, 23: 347-360.

Grossberg, S. (1976). Adaptive pattern classification and universal recording, II. Feedback expectation, olfation and illusions. *Biological Cybernetics*, 23: 187-202.

Guazzelli, A. (1994). *Aprendizagem em sistemas hibridos (Learning in hybrid systems).* MSc. dissertation, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil. 130 pages.

Guazzelli, A. & Leao, B. F. (1994). Incorporating Semantics to ART. *Proceedings IEEE Conference on Neural Networks,* 1726-1731, Orlando, FL. New York, NY: IEEE Press.

Hammond, K. (1986). CHEF: A model of case-based planning. *Proceedings of the AAAI.* Cambridge, MA: AAAI Press/MIT Press.

Harmon, P. & King, D. (1985). *Artificial Intelligence in Business.* New York, NY: John Wiley.

Hayes-Roth, I. (1983). *Building Expert Systems*. Reading, MA: Addison-Wesley.

Hendler, J. A. (1989). Marker-passing over microfeatures: towards a hybrid symbolic/connectionist model. *Cognitive Science*, 13: 79-106.

Henessy, D. & Hinkle, D. (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert*, 7(5):21-26.

Hinrichs, T. R. (1988). Towards an architecture for open world problem solving. In Kolodner, J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, Clearwater Beach, FL. San Mateo, CA: Morgan Kaufmann.

Hinrichs, T. R. (1989). Strategies for adaptation and recovery in a design problem solver. In Hammond, K. J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, Pensacola Beach, FL. San Mateo, CA: Morgan Kaufmann.

Honavar, V. (1995). Symbolic artificial intelligence and numeric artificial neural networks: towards a resolution of the dichotomy. In Sun, R. & Bookman, L. A. (eds.), *Computational Architectures Integrating Neural and Symbolic Processes: a Perspective on the State of the Art*, 351-388. Dordrecht, Netherlands: Kluwer Academic.

Hopfield, J. J. (1979). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science USA*, 2554-2558.

Hudson, D. L., Banda, P. W., Cohen, M. E., Blois, M. S. (1992). Medical diagnosis and treatment plans derived from a hybrid expert system. In Kandel, A. & Langholz, G. (eds.), *Hybrid Architectures for Intelligent Systems*, 329-344. Boca Raton, FL: CRC Press.

Hull, R. & King, R. (1987). Semantic database modeling: survey, applications, and research issues. *ACM Computing Surveys*, 19: 201-260.

Hunter, L. (1989). Finding paradigm cases or when is a case worth remembering? In Hammond, K. J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, 57-61, Pensacola Beach, FL. San Mateo, CA: Morgan Kaufmann.

Jackson, P. (1986). *Introduction to Expert Systems.* Reading, MA: Addison-Wesley.

Kintch, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. *Psychological Review*, 95: 163-182.

Klein, G. A. & Calderwood, R. (1988). How do people use analogues to make decisions. In Kolodner, J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, Clearwater Beach, FL. San Mateo, CA: Morgan Kaufmann.

Knaus, R. (1992). Representing expert knowledge in neural nets. In Kandel, A. & Langholz, G. (eds.), *Hybrid Architectures for Intelligent Systems*, 345-356. Boca Raton, FL: CRC Press.

Kohonen, T. (1988). The neural fonetic typewriter. *IEEE Computer*, 21(3): 11-22.

Kolodner, J. (1983a). Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7: 243-280.

Kolodner, J. (1983b). Reconstructive memory: a computer model. *Cognitive Science*, 7: 281-328.

Kolodner, J. & Simpson, R. L. (1989). The MEDIATOR: Analysis of an early case-based problem solver. *Cognitive Science*, 13(4): 507-549.

Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann.

Kopeikina, L., Brandau, R. & Lemmon, A. (1988). Case-based reasoning for continuos control. In Kolodner, J. (ed.). *Proceedings of the Case-Based Reasoning Workshop*, Clearwater Beach, FL. San Mateo, CA: Morgan Kaufmann.

Koton, P. (1988). Reasoning about evidence in causal explanations. *Proceedings of AAAI*, 256-261, Saint Paul, Minnesota. Cambridge, MA: AAAI Press.

Krovvidy, S. & Wee, W. G. (1992). An Intelligent hybrid system for wastewater treatment. In Kandel, A. & Langholz, G. (eds.), *Hybrid Architectures for Intelligent Systems*, 345-356. Boca Raton, FL: CRC Press.

Kurzweil, R. (1990). *The Age of Intelligent Machines*, pages 569-570. Cambridge, MA: MIT

Press.

Lange, T. E. (1992). Hybrid connectionist models: temporary bridges over the gap between the symbolic and the subsymbolic. In Dinsmore, J. (ed.), *The connectionist and the symbolic paradigm: closing the gap*, 237-289. Hillsdale, NJ: Erlbaum.

Leao, B. F. (1988). *Construcao da base de conhecimento de um sistema especialista de apoio ao diagnostico de cardiopatias congenitas* (The construction of an expert system knowledge base for the diagnostic support of congenital heart diseases). PhD thesis, University of Sao Paulo, Sao Paulo, Brazil. 230 pages.

Leao, B. F. & Rocha, A. F. (1990). Proposed methodology for knowledge acquisition: a study on congenital heart disease diagnosis. *Methods of Information in Medicine*, 29: 30-40.

Leao, B. F., Timmers, T., Van der Lei, D., Mulligan, E. M. (1990). Heartview - an object-oriented knowledge base to support clinical research in cardiology. In *Proceedings of Computers in Cardiology*, 88-92, Chicago, IL. New York, NY: IEEE Press.

Leao, B. F. & Reategui, E. B. (1993). Hycones: a hybrid connectionist expert system. In *Proceedings of the Symposium on Computer Applications in Medical Care*, Washington, DC. New York, NY: IEEE Press.

Leao, B. F., Reategui, E. B. & Guazzelli, A. (1994). Hycones II: a tool to develop hybrid connectionist expert systems. *Proceedings of the International Congress on Biomedical Engineering*, Rio de Janeiro, Brazil.

Leao, B. F., Reategui, E. B., Guazzelli, A. & Mendonca, E. (1994). Hybrid systems - a promising solution to better decision support tools. *Proceedings Medinfo - the VIII World Conference on Medical Informatics*, Vancouver, Canada.

Lebowitz, M. (1983). Generalization from natural language text. *Cognitive Science* 1(7).

Lebowitz, M. (1987). Eperiment with incremental concept formation: UNIMEN. *Machine Learning*, 2(2): 103-138.

Lenz, M. (1993). Cabata - a hybrid CBR system. In Althoff, K-D., Richter, K. & Wess, S. (eds.). *Proceedings of the First European Workshop on Case-Based Reasoning*, 204-

209, Kaiserslautern, Germany.

Lim, J. H., Lui, H. C., Tan, A. H. & Teh, H. H. (1991). A connectionist case-based diagnostic expert system that learns incrementally. *Proceedings of the International Joint Conference on Neural Networks*, 1693-1698.

Lin, C. & Hendler, J. (1995). Examining a hybrid connectionist/symbolic system for the analysis of ballistic signals. In Sun, R. & Bookman, L. A. (eds.), *Computational Architectures Integrating Neural and Symbolic Processes: a Perspective on the State of the Art*, 319-348. Dordrecht, Netherlands: Kluwer Academic.

Long, J. M. et al. (1991). Automating the discovery of causal relationships in a medical records database. In Piatetsky-Shapiro, G. & Frawley W. J. (eds.), *Knowledge Discovery in Databases*, 465-476. Menlo Park, CA: AAAI Press.

Macchion, D. & Vo, D. (1993). A hybrid KBS for technical diagnosis learning and assistance. In Althoff, K-D., Richter, K. & Wess, S. (eds.), *Proceedings of the First European Workshop on Case-Based Reasoning*, 307-312, Kaiserslautern, Germany.

Magaldi, R. V. (1994). CBR for troubleshooting aircraft on the flightline. In *Proceedings IEE Colloquium on Case-Based Reasoning: Prospects for Applications.* Digest No: 1994/057. pp. 6/1-6/9.

Mattos, N. M. (1991). *An approach to knowledge base management.* Berlin: Springer Verlag.

McCulloch, W. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5: 115-137.

Machado, R. J. (1989). Handling knowledge in high order neural networks: the combinatorial neural model. Technical report CCR076, IBM Rio Scientific Center, Rio de Janeiro, Brazil.

Machado, R. J., Rocha, A. F. & Leao, B. F. (1990). Calculating the mean knowledge representation from multiple experts. In Fedrezzi, M. & Kacprzkyk, J. (eds.), *Multiperson decision making models using fuzzy sets and possibility theory*, 113-127. Dordrecht, Netherlands: Kluwer Academic.

Machado, R. J. & Rocha, A. F. (1990). The combinatorial neural network: a connectionist model for knowledge based systems. In Bouchon-Meunier, B., Yager, R. R. & Zadeh, L. A. (eds.), *Uncertainty in knowledge bases*. Berlin: Springer Verlag.

Machado, R. J. et al. (1991). NEXT: *The Neural Expert Tool.* Technical report CCR120, IBM Rio Scientific Center, Rio de Janeiro, Brazil.

Machado, R. J. & Rocha, A. F. (1992). A hybrid architecture for fuzzy connectionist expert systems. In Kandel, A. & Langholz, G. (eds.). *Hybrid architectures for intelligent systems*, 135-152. Boca Raton, FL: CRC Press.

Machado, R. J., Rocha, A. F. & Denis, F. A. R. M. (1992). Evolutive Learning in Neural Networks. *Proceedings of IEEE International Conference on Fuzzy Systems*, 762-767, San Diego, CA.

Malek, M. (1995). A connectionist indexing approach for CBR systems. In Veloso, M. & Aamodt, A. (eds.), *Case-Based Reasoning: Research and Development. First International Conference, ICCBR-95*, 520-527, Sesimbra, Portugal. Berlin, Germany: Springer Verlag.

Manago, M. et al. (1993). Induction and Reasoning from Cases. In Althoff, K-D., Richter, K. & Wess, S. (eds.), *Proceedings of the First European Workshop on Case-Based Reasoning*, 313-318, Kaiserslautern, Germany.

McLeish et al. (1991). Discovery of medical diagnostic information: an overview of methods and results. In Piatetsky-Shapiro, G. & Frawley W. J. (eds.), *Knowledge Discovery in Databases,* 477-490. Menlo Park, CA: AAAI Press.

Medsker, L. R. & Bailey, D. L. (1992). Models and guidelines for integrating expert systems and neural networks. In Kandel, A. & Langholz, G. (eds.), *Hybrid architectures for intelligent systems*, 329-344. Boca Raton, FL: CRC Press.

Michalsky, R. S. & Chilausky, R. L. (1980). Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International*

Journal of Policy Analysis and Information Systems, 4(2): 156-161.

Michalski, R. S., Carbonell, J. G. & Mitchell, T. M. (1983). *Machine Learning: an Artificial Intelligence Approach, vol. 1.* Palo Alto, CA: Tioga.

Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (1994). Introduction (Chapter 1). In Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (eds.), *Machine learning, neural and statistical classification*, 1-5. Chichester, England: Ellis Horwood.

Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (1994). Conclusions (Chapter 11). In Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (eds.), *Machine learning, neural and statistical classification*, 213-227. Chichester, England: Ellis Horwood.

Miller, G. A. (1956). The Magical Number 7, plus or minus 2: Some limits on our capacity for processing information. *Psychology Review*, 63: 81-97.

Miller, R. A. et al. (1986). Internist-I: an experimental computer-based diagnostic consultant for general internal medicine. In Reggia, J. A. & Stanley, T. (eds.), *Computer-assisted medical decision making*, vol. 2, 139-158. New York, NY: Springer Verlag.

Minsky, M. & Papert, S. (1969). *Perceptrons - An Introduction to Computational Geometry.* Cambridge, MA: MIT Press.

Minsky, M. A. (1975). A framework for representing knowledge. In Winston, P. H. (ed.), *The Psychology of Computer Vision.* New York, NY: McGraw-Hill.

Murphy, P. M. & Aha, D. W. (1994). *UCI Repository of machine learning databases* [Machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science.

Myllymaki, P. & Tirri, H. (1993). Massively parallel case-based reasoning with probabilistic similarity metrics. In Althoff, K-D., Richter, K. & Wess, S. (eds.), *Proceedings of the First European Workshop on Case-Based Reasoning*, 48-53, Kaiserslautern, Germany.

Newell, A. & Simon, H. A. (1993). GPS, a program that simulates human thought. In Feigembaum, E. & Feldman, J. (eds.), *Computers and Thought*, 279-293. New York, NY: McGraw Hill.

Nilsson, N. J. (1990). *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga.

Owens, D. K. et al. (1990). Medical decision making: probabilistic medical decision making. In Shortlife, E. H. et al. (eds.), *Medical Informatics: computer applications in health care*, 70-116. Reading, MA: Addison-Wesley.

Porter, B. W., Bareiss, R. & Holte, R. C. (1990). Concept learning and heuristic classification in weak theory domains. *Artificial Intelligence*, 45(1-2): 229-263.

Quinlan, J. R. (1993). *C4.5 Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. (1986). Induction of decision trees, *Machine Learning*, 1(1): 81-106.

Reategui, E. B. & Leao, B. F. (1993). Integrating neural networks with the formalism of frames. In Grossberg, S. (ed.), *Proceedings of the World Congress on Neural Networks*, Portland, OR. Northvale, NJ: Erlbaum.

Reategui, E. B. (1993). *Um Modelo para Sistemas Especialistas Conexionistas Hibridos (A model for hybrid connectionist expert systems).* MSc dissertation, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil.

Reategui, E. B. & Campbell, J. A. (1995). A classification system for credit card transactions. In Haton, J-P., Keane, M. & Manago, M. (eds.), *Advances in Case-Based Reasoning: Second European Workshop EWCBR-94*, 280-291, Chantilly, France. Berlin, Germany: Springer Verlag.

Reategui, E. B., Campbell, J. & Borghetti, S. (1995). Using a neural network to learn general knowledge in a case-based system. In Veloso, M. & Aamodt, A. (eds.), *Case-Based Reasoning: Research and Development. First International Conference, ICCBR-95*, 528-537, Sesimbra, Portugal. Berlin, Germany: Springer Verlag.

Reategui, E. B., Campbell, J. & Leao, B. F. (1996a). A case-based model that combines general and specific knowledge in reasoning. To appear in: *Applied Intelligence.*

Reategui, E., Campbell, J. A., Leao, B. F. (1996b). Combining a Neural Network with Case-Based Reasoning in a Diagnostic System. To appear in *Artificial Intelligence in*

Medicine Journal - special issue devoted to Case-Based Reasoning.

Riesbeck, C. K. & Schank, R. C. (1989). *Inside case-based reasoning*. Northvale, NJ: Erlbaum.

Rissland, L. R. & Ashley, K. D. (1988). Credit assignment and the problem of competing factors in case-based reasoning. In Kolodner, J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, 327-344, Clearwater Beach, FL. San Mateo, CA: Morgan Kaufmann.

Rissland, E. L. & Skalak, D. B. (1989). Combining case-based and rule-based reasoning: a heuristic approach. *Proceedings of International Joint Conference on Artificial Intelligence*, 524-530, Detroit, USA. San Mateo, CA: Morgan Kaufmann.

Romaniuk, S. G. & Hall, L. O. (1989). *FUZZNET: a fuzzy connectionist expert system*, Technical report CCR120. Department of Computer Science and Engineering, University of South Florida, FL.

Rosenblatt, F. (1962). *Principles of Neurodynamics*. New York, NY: Spartan Books.

Ross, B. H. (1989). Some psychological results on case-based reasoning. In Hammond, K. J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, 318-323, Pensacola Beach, FL. San Mateo, CA: Morgan Kaufmann.

Rumelhart, D. E, McClelland, J. L. & PDP Research Group (1986). *Parallel Distributed Processing: explorations in the microstructures of cognition.* Cambridge, MA: MIT Press.

Rumelhart, D. E, Hinton, G. E. & McClelland, J. L. (1986). Learning internal representations by error propagation. In Rumelhart, D. E, McClelland, J. L. & PDP Research Group (eds.), *Parallel Distributed Processing: explorations in the microstructures of cognition*, vol. 1, Cambridge, MA: MIT Press.

Samad, T. (1988). Towards Connectionist Rule-Based Systems. *Proceedings of the Conference on Neural Information Processing Systems*, vol. 2, 525-531.

Sammut, C. (1994). Knowledge Representation (Chapter 12). In Michie, D., Spiegelhalter, D.

J., Taylor, C. C. (eds.). *Machine learning, neural and statistical classification*, 229-245. Chichester, England: Ellis Horwood.

Schank, R. C. (1982). *Dynamic Memory: a Theory of Understanding and Learning in Computers and People.* Cambridge, England: Cambridge University Press.

Simpson, P. (1990). *Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations.* Oxford, England: Pergamon Press.

Skalak, D. B. (1989). Options for controlling mixed paradigm systems. In Hammond, K. J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, 318-323, Pensacola Beach, FL. San Mateo, CA: Morgan Kaufmann.

Smyth, B. & Keane, M. T. (1995). Experiments on adaptation-guided retrieval in case-based design. In Veloso, M. & Aamodt, A. (eds.), *Case-Based Reasoning: Research and Development. First International Conference, ICCBR-95*, 313-324, Sesimbra, Portugal. Berlin, Germany: Springer Verlag.

Sternberg, R. J. (1984). Reasoning, problem solving and intelligence. In Sternberg, R. J. (ed.), *The Handbook of Human Intelligence*, 225-307. Cambridge, England: Cambridge University Press.

Strube, G., Enzinger, A., Janetzko, D. & Knauff, M. (1995). Knowledge Engineering for CBR systems from a cognitive science perspective. In Veloso, M. & Aamodt, A. (eds.), *Case-Based Reasoning: Research and Development. First International Conference, ICCBR-95*, 548-558, Sesimbra, Portugal. Berlin, Germany: Springer Verlag.

Sun, R. (1995a). An introduction on symbolic processing in neural networks. In Sun, R. & Bookman, L. A. (eds.), *Computational Architectures Integrating Neural and Symbolic Processes: a Perspective on the State of the Art,* 1-20. Dordrecht, Netherlands: Kluwer Academic.

Sun, R. (1995b) A Two-level architecture for structuring knowledge for commonsense reasoning. In Sun, R. & Bookman, L. A. (eds.), *Computational Architectures Integrating Neural and Symbolic Processes: a Perspective on the State of the Art*, 247-281.

Dordrecht, Netherlands: Kluwer Academic.

Surma, J. & Vanhoof, K. (1995). Integrating Rules and Cases for the Classification Task. In Veloso, M. & Aamodt, A. (eds.), *Case-Based Reasoning: Research and Development. First International Conference, ICCBR-95*, 325-334, Sesimbra, Portugal. Berlin, Germany: Springer Verlag.

Sycara, K. (1988). Using case-based reasoning for plan adaptation and repair. In Kolodner, J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, Clearwater Beach, FL. San Mateo, CA: Morgan Kaufmann.

Theoto, M. R. et al. (1987). The fuzzy decodings of educative texts. *Fuzzy Sets and Systems*, 23: 331-345.

Thrift, P. (1989). A neural network model for case-based reasoning, In Hammond, K. J. (ed.), *Proceedings of the Case-Based Reasoning Workshop*, 334-337, Pensacola Beach, FL. San Mateo, CA: Morgan Kaufmann.

Thrun, S. B. et al. (1991). *The MONK's Problems: A Performance Comparison of Different Learning Algorithms*, Technical report CMU-CS-91-197. Carnegie Mellon University.

Tirri, H. (1995). Replacing the pattern matcher of an expert system with a neural network. In S. Goonatilake & Khebbal, S. (eds.), *Intelligent Hybrid Systems*, 47-61. New York, NY: John Wiley.

Touretzky, D. & Hinton, G. (1987). Symbols among neurons. *Proceedings of the 9th International Joint Conference of Artificial Intelligence*, 238-243. San Mateo, CA: Morgan Kaufmann.

Uehara, K., Tanizawa, M. & Maekawa, S. (1993). PBL: Prototype-based learning algorithm. In Althoff, K-D., Richter, K. & Wess, S. (eds.), *Proceedings of the First European Workshop on Case-Based Reasoning*, 160-165, Kaiserslautern, Germany.

Uthurusamy, R., Fayyad, U. M. & Spangler, S. (1991). Learning useful rules from inconclusive data. In Piatetsky-Shapiro, G. & Frawley W. J. (eds.), *Knowledge Discovery in Databases*, 141-157. Menlo Park, CA: AAAI Press.

Vanlehn, K. (1989). Problem solving and cognitive skill acquisition. In Posner, M. I. (ed.), *Foudations of Cognitive Science*, 527-579. Cambridge, MA: MIT Press.

Watson, I. & Marir, F. (1994). Case-based reasoning: a review. *The Knowledge Engineering Review*, 9(4): 327-354.

Winston, P. H. (1993). *Artificial Intelligence.* Reading, MA: Addison Wesley.

Yang, Q. & Bhargava, V. K. (1990). Building expert systems by a modified perceptron network with rule-transfer algorithms. In *Proceedings of the International Joint Conference on Neural Networks, 77-82.*

Zeleznikow, J., Hunter, D. & Vossos, G. (1993). Integrating rule-based and case-based reasoning with information retrieval: the IKBALS project. In Althoff, K-D., Richter, K. & Wess, S. (eds.), *Proceedings of the First European Workshop on Case-Based Reasoning*, 341-346, Kaiserslautern, Germany.