# Uma Abordagem Híbrida Baseada em Casos e Redes Neurais. Uma Aplicação: Escolha e Configuração de Modelos de Redes Neurais

***Ricardo Barz Sovat***

Orientador :

Prof. Dr. ***André Carlos Ponce de Leon F. de Carvalho***

Co-orientadora:

Profª. Drª. ***Sandra Maria Aluísio***

*Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - USP, como parte dos requisitos para a obtenção do título de Doutor em Ciências - Área de Ciências de Computação e Matemática Computacional.*

**USP - São Carlos**
**Maio de 2002**

**(Versão Revisada)**

Data da Defesa: 07/03/2002

Visto do Orientador:

Trabalho realizado com o apoio da FAPESP

Uma Abordagem Híbrida Baseada em Casos e Redes Neurais.
Uma Aplicação: Escolha e Configuração de Modelos de Redes Neurais

# Resumo


Nesta tese, é pesquisada a integração entre dois paradigmas da área de Inteligência Artificial, Raciocínio Baseado em Casos e Redes Neurais Artificiais. Essa pesquisa dá-se em dois sentidos. Primeiro, o estudo da aplicação da metodologia de Raciocínio Baseado em Casos ao problema da escolha e configuração de um modelo de Rede Neural Artificial. Em segundo lugar, a viabilidade da introdução de uma Rede Neural Artificial no interior do ciclo de funcionamento de um sistema baseado em casos.

As soluções para o problema de escolha e configuração de um modelo de Rede Neural ainda possuem, até hoje, uma forte componente empírica. Não existe um conhecimento formalizado disponível que forneça suporte a um processo único de implementação destes sistemas, ditos conexionistas. A qualidade da solução depende em muito da habilidade do projetista em ajustar um conjunto de diversos parâmetros envolvidos.

A metodologia de Raciocínio Baseado em Casos, por sua vez, fundamenta-se na idéia de que um especialista eficiente não é um processador de regras, mas um acumulador de experiências práticas, bem e mal sucedidas. Desta forma, ela torna-se bastante adequada à aplicação em domínios em que o conhecimento é mais difuso, ou seja, não pode ser facilmente explicitado.

A partir destas observações, é proposta a representação do problema como uma tarefa tipicamente de projeto (*design*) e estabelecida uma estratégia para aplicar a metodologia em sua solução.

No outro sentido, a escolha da melhor solução, dentro da metodologia de Raciocínio Baseado em Casos, depende de bons processos que permitam a transformação de uma solução anterior em uma solução adequada ao problema atual. Esses processos podem beneficiar-se, conforme é mostrado ao longo do trabalho, de uma boa capacidade de generalização de um conhecimento adquirido. Na maioria dos sistemas existentes, essas transformações, ou adaptações, são executadas através de regras de produção. Essas regras por sua vez exigem também um grau de aquisição de conhecimento em domínios nem sempre bem estruturados.

Redes Neurais Artificiais possuem como ponto forte a capacidade de aprender a partir de exemplos, extrair características intrínsecas de conjuntos de dados e generalizar esse conhecimento adquirido. Essa capacidade as credencia como boas alternativas para substituição de sistemas baseados em regras. O que pode ser considerado um ponto fraco das Redes Neurais, sua carência de justificativas para, por exemplo, associações ou previsões efetuadas, não constitui um empecilho para sua intodução neste ponto específico do ciclo de Raciocínio Baseado em Casos.

Com base nestas premissas, este trabalho sugere uma abordagem híbrida neuro-simbólica como mecanismo de recuperação e adaptação de casos desse ciclo.

Para servir como ferramenta de testes, foi também implementado um ambiente de desenvolvimento de sistemas de Raciocínio Baseado em Casos.

Advisor
**André Carlos P. L. Ferreira de Carvalho**

Author
**Ricardo Barz Sovat**

A Hybrid Approach Based on Cases and Neural Networks.
An Application: Neural Network Models Choice and Configuration.

# Abstract

In this thesis, the integration of two Artificial Intelligence paradigms, Case-Based Reasoning and Artificial Neural Networks, is studied. The research is performed in two different directions. First, the study of applying the Case-Based Reasoning methodology to the problem of choosing and configuring an Artificial Neural Network model. In second place, the feasibility of introducing an Artificial Neural Network inside of a case-based system working cycle.

The solutions to the problem of choosing and configuring an Artificial Neural Network model have a strong empirical component. There is no available formalized knowledge that provides substance for an unified implementation process of those systems, known as connectionist. The solution quality depends upon the designer skill in adjusting a set of several related parameters.

The Case-Based Reasoning methodology has its fundaments on the idea that an efficient expert is not a rule processor, but a collector of practical experiences, well succeeded or not. So, the methodology becomes very sound to be applied to domains where the knowledge is more diffuse and it is difficult to make it explicit.

From those observations, it is proposed the problem representation as a typical design task and it is established a strategy to apply the methodology in its solution.

In the other direction, the choice of the best solution, inside the Case-Based Reasoning methodology, depends upon the existence of good processes that allow the transformation of a former solution into an adequate solution to the present problem. Those processes can take profit, as is shown along the work, of a good generalization capacity of the acquired knowledge. In most of the actual systems, those transformations, or adaptations, are accomplished by production rules. Those rules also demand a high degree of knowledge acquisition in domains not always well structured.

Artificial Neural Networks have as a strong characteristic the ability of learning from examples, extract intrinsic features from datasets and to generalize this acquired knowledge. This ability gives them credentials to be good options in substituting rule based systems. What could be considered a weak characteristic of the Neural Networks, its leak of justifications to make its associations or predictions, does not constitute a barrier to its introduction in this specific point of the Case-Based Reasoning cycle.

Based on those premises, this work suggests a neurosymbolic hybrid approach as a mechanism of retrieving and adapting cases inside this cycle.

In order to provide a testing tool, it was also created a Case-Based Reasoning development environment.

**À memória de meu pai, Dick Sovat, meu incentivador e orientador em todos os estudos.**

# Agradecimentos

Ao Professor Dr. André de Carvalho, meu orientador nesta tese, não só por seus valiosos conselhos e opiniões de ordem técnica, mas também pelo exemplo constante de como é possível viver rodeado pelas máquinas sem descuidar de valores que estão entre os melhores do ser humano: amizade, honestidade, paciência, otimismo e dedicação.

À Professora Drª Sandra Maria Aluísio, co-orientadora deste trabalho, por participar nessa minha incursão pelo "mundo" do simbolismo, pelo apoio e estímulos constantes e, sobretudo, por me ensinar como se escreve uma tese, como organizar uma variedade de idéias e informações e passá-las para o papel de forma elegante. Espero que eu possa adquirir essa sua habilidade.

À Maria Cristina, pela mágica de conseguir ser esposa, mãe, pesquisadora e ainda me ajudar na confecção desta tese, material e emocionalmente, sem perder sua habitual dedicação, eficiência e carinho em todas suas atividades.

A minhas filhas, Ana Carolina e Maria Luísa, por seu carinho e pela paciência, tão difícil para as crianças, quando precisam ficar ao mesmo tempo longe do pai... e do computador.

A minha mãe, Zilda, pelo incentivo e por sua constante demonstração de amor e dedicação. Como em todos os momentos de minha vida, foi uma presença reconfortante e fortalecedora.

Ao restante de minha família e aos meus amigos, pela compreensão pelo meu "sumiço".

À Professora Drª Roseli Francelin Romero, pela oportunidade de participar na proposta e elaboração dos exercícios do curso de Redes Neurais e pela cessão dos trabalhos executados.

Ao Professor Dr. Marinho Gomes de Andrade Filho, pela ajuda no momento de "enfrentar" e compreender a estatística necessária para este trabalho.

A todos os professores e amigos do LABIC, pela demonstração viva de como reunir um grupo produtivo com camaradagem e cooperação, construindo um dos melhores ambientes que já conheci em minha vida profissional.

À Marília, Laura, Sandra, Adriana, Tatiana e Beth, pela correta e eficiente condução não só dos trâmites burocráticos, mas também daqueles problemas do dia-a-dia que sempre acabam em suas mãos.

À FAPESP, Fundação de Apoio à Pesquisa do Estado de São Paulo, pelo suporte acadêmico e financeiro que permitiram a minha dedicação a esta tese e pelo exemplo de seriedade e organização tão importantes para a administração pública em nosso país.

# Conteúdo

**Páginas Iniciais**

Página de Título ..... i
Resumo ..... iii
Abstract ..... v
Dedicatória ..... vii
Agradecimentos ..... ix
Conteúdo ..... xi
Lista de Figuras ..... xiii
Lista de Tabelas ..... xiv

**INTRODUÇÃO ..... 1**

1.1 MOTIVAÇÃO ..... 1
1.2 OBJETIVO ..... 3
1.3 CARACTERÍSTICAS DA APLICAÇÃO ..... 4
1.4 ORGANIZAÇÃO ..... 6

**RACIOCÍNIO BASEADO EM CASOS ..... 9**

2.1 DESCRIÇÃO DA METODOLOGIA ..... 9
2.2 FERRAMENTAS EXISTENTES ..... 10
2.3 CICLO BÁSICO DE FUNCIONAMENTO ..... 12
2.4 ADAPTAÇÃO DE CASOS ..... 14
*2.4.1 Métodos de Substituição ..... 14*
*2.4.2 Métodos de Transformação ..... 15*
2.5 CONSIDERAÇÕES FINAIS ..... 16

**UMA ABORDAGEM HÍBRIDA ..... 17**

3.1 ABORDAGENS ANTERIORES ..... 17
3.2 CARACTERÍSTICAS DO PROBLEMA ..... 21
3.3 ABORDAGEM PROPOSTA ..... 23
*3.3.1 Memória Associativa ..... 25*
*3.3.2 Representação Adotada ..... 26*
*3.3.3 Problemas e Soluções ..... 28*
*3.3.4 Funcionamento Básico ..... 29*
3.4 CONSIDERAÇÕES FINAIS ..... 31

**RABECA: O AMBIENTE DE DESENVOLVIMENTO** ..... **33**

4.1 PRINCIPAIS COMPONENTES DO AMBIENTE ..... 35

*4.1.1 Indexação empregada* ..... *40*

*4.1.2 Critérios de Similaridade* ..... *42*

4.2 CONSIDERAÇÕES FINAIS ..... 44

**A MONTAGEM DA BASE DE CASOS** ..... **45**

5.1 ESTRUTURA DE UM CASO ..... 45

5.2 PRINCIPAIS PARÂMETROS ..... 46

5.3 ESCOLHA DE ATRIBUTOS ..... 48

*5.3.1 Índices Estatísticos* ..... *49*

*5.3.2 Atributos Índice* ..... *52*

*5.3.3 Organização da Memória* ..... *52*

5.4 REGRAS ..... 52

5.5 GENERALIZAÇÕES ..... 54

5.6 OS CASOS ..... 54

5.7 CONSIDERAÇÕES FINAIS ..... 62

**METODOLOGIA DE AVALIAÇÃO** ..... **63**

6.1 O QUE AVALIAR ? ..... 64

6.2 CRITÉRIOS ..... 64

6.3 SISTEMA HÍBRIDO ..... 65

*6.3.1 Metodologia de Avaliação do Sistema Híbrido* ..... *66*

6.4 SISTEMA FINAL ..... 66

6.5 CONSIDERAÇÕES FINAIS ..... 67

**UM ESTUDO DE CASO: CLASSIFICAÇÃO DE PADRÕES** ..... **69**

7.1 DESCRIÇÃO DO EXPERIMENTO ..... 69

7.2 RESULTADOS ..... 72

7.3 CONSIDERAÇÕES FINAIS ..... 73

**CONCLUSÃO E CONTRIBUIÇÕES** ..... **77**

8.1 CONTRIBUIÇÕES ..... 77

8.2 LIMITAÇÕES ..... 79

8.3 TRABALHOS FUTUROS ..... 80

**LISTAGEM DA BASE DE CASOS** ..... **83**

**SINTAXE DA LINGUAGEM CASL** ..... **95**

**DESCRIÇÃO FUNCIONAL DO RABECA** ..... **101**

**REFERÊNCIAS BIBLIOGRÁFICAS** ..... **115**

# Lista de Figuras

FIGURA 2.1 – O CICLO BÁSICO DE FUNCIONAMENTO DE RBC ..... 13
FIGURA 3.1 - CLASSIFICAÇÃO DE SISTEMAS NEURO-SIMBÓLICOS INTEGRADOS ..... 18
FIGURA 3.2 - MODOS DE INTEGRAÇÃO NEURO-SIMBÓLICOS HÍBRIDOS ..... 21
FIGURA 3.3 – FUNCIONAMENTO BÁSICO DA MEMÓRIA ASSOCIATIVA. ..... 28
FIGURA 4.1 - DIAGRAMA DE INTERLIGAÇÃO DOS COMPONENTES DO RABECA ..... 35
FIGURA 4.2 - TRECHO INICIAL DE SCRIPT EM CASL ..... 36
FIGURA 4.3 - TRECHO DE *SCRIPT*, SEÇÃO DE MODIFICAÇÕES ..... 37
FIGURA 4.4 - REGRAS DE PRÉ-PROCESSAMENTO ..... 38
FIGURA 4.5 – INSTÂNCIA DE CASO E REGRAS DE REPARO LOCAIS ..... 40
FIGURA 4.6 - TELA BÁSICA DO SISTEMA ..... 41
FIGURA 5.1 – PERFIS DE DISTRIBUIÇÕES QUANTO À ASSIMETRIA ..... 51
FIGURA 5.2 - PERFIS DE DISTRIBUIÇÃO QUANTO AO ACHATAMENTO ..... 51
FIGURA 5.3 -CAMPOS DE UM CASO DE RNAS ..... 54
FIGURA 5.4 – TELA DE CONSULTA À BASE DE CASOS ..... 55
FIGURA 5.5 - ATRIBUTOS ÍNDICE ..... 55
FIGURA 5.6 - ATRIBUTOS DO CASO ..... 56
FIGURA 5.7 - REGRAS DE PRÉ-PROCESSAMENTO ..... 57
FIGURA 5.8 - REGRAS DE REPARO ..... 57
FIGURA 5.9 - AS REGRAS EXIBIDAS DENTRO DO RABECA ..... 58
FIGURA 5.10 - GENERALIZAÇÕES EMPREGADAS ..... 58
FIGURA 5.11 - TELA DE GENERALIZAÇÕES : ABSTRAÇÕES ..... 59
FIGURA 5.12 - TELA DE GENERALIZAÇÕES : INTERVALOS SIMILARES ..... 59
FIGURA 5.13 - CASO RECUPERADO ..... 60
FIGURA 5.14 - DESCRIÇÃO DO CASO NA BASE ..... 61
FIGURA 5.15 - TELA DA ETAPA DE REVISÃO ..... 61
FIGURA 7.1 - DIAGRAMA DE BLOCOS DO EXPERIMENTO ..... 71
FIGURA 7.2 - CASO RECUPERADO NO EXPERIMENTO ..... 72
FIGURA 7.3 - TELA DO CASO NOVO (ADAPTADO) ..... 74
FIGURA C.1 - DIAGRAMA DE FLUXO DOS PRINCIPAIS MÓDULOS DE *SOFTWARE* ..... 101

# Lista de Tabelas

**TABELA 7.1 - COMPARAÇÃO DOS CASOS** ..... 73

**TABELA 7.2 – COMPARAÇÃO DOS DESEMPENHOS DAS RNAS** ..... 73

# Capítulo 1

# Introdução

Neste capítulo, é apresentada uma descrição geral das idéias básicas, de ordem prática ou não, que motivaram esta tese. Para tal, antes de tudo, são mencionados os pontos teóricos iniciais que inspiraram o trabalho. Em seguida, são indicados o tipo de desenvolvimento desejado e delineado o método de trabalho escolhido. Finalmente, será descrita a organização do texto completo.

## 1.1 Motivação

Esta pesquisa consiste em estudar as possibilidades de reunião de dois paradigmas aplicados com sucesso na área de Inteligência Artificial: Redes Neurais Artificiais (RNAs) [Haykin 1999] e Raciocínio Baseado em Casos (RBC) [Kolodner 1993].

Este trabalho enquadra-se numa seqüência de estudos relacionados a Raciocínio Baseado em Casos, anteriormente iniciada no âmbito do Laboratório de Inteligência Computacional do ICMC / USP (LABIC) [Lopes 1995] [Fabiunke et al. 1997] [Milaré, Carvalho 1998].

Inicialmente, deve-se explicar que a idéia de construir um sistema híbrido é apoiada na observação de que sistemas inteligentes que sejam bem sucedidos em aplicações práticas reais devem possuir, em grande parte dos casos, além de um suporte de *software* robusto, características que reunam o melhor de dois mundos, o simbolismo e o conexionismo.

É interessante notar que tal comportamento é, de certo modo, compatível com as soluções encontradas na natureza, em que a fisiologia de sistemas nervosos apresenta clara diferenciação estratégica conforme sua funcionalidade, o que sugere que, quanto mais adaptável seja um organismo, maior variedade estrutural ele apresente. O sistema nervoso central humano, por exemplo, apesar de ser basicamente um conjunto extremamente complexo de estruturas neuronais, apresenta características típicas de abordagens cognitivas no que se relaciona ao processo de atenção. Alguns aspectos do processo de visão são fortemente

influenciados por experiências anteriores, também [Popper, Eccles 1977][Levine 1991]. A observação dos sistemas biológicos sugere um conjunto funcionalmente heterogêneo que possui uma base fisiológica comum, que são os neurônios.

Um sistema inteligente, uma vez que se proponha a replicar um comportamento encontrado em processos biológicos, possui uma boa chance de ter seu desempenho melhorado uma vez que busque também copiar esta aparente estrutura modular observada.

Por sua vez, a escolha da hibridização envolvendo especificamente RBC e RNAs, é proveniente da observação de similaridades entre o funcionamento de ambos. Tanto RBC quanto RNAs aprendem a partir de exemplos. Ainda que as RNAs não costumem ser consideradas como exemplos de Aprendizado Baseado em Instâncias, enquanto RBC por vezes seja citado como tal, efetivamente essa é uma característica comum aos dois paradigmas.

Suas diferenças de operação provêm justamente do tipo de informação mais apropriada para seu manuseio. Enquanto RNAs processam sempre valores numéricos e são eficientes em extrair características implícitas, RBC relaciona-se melhor com descrições de entidades do ambiente do domínio. Outra diferença notável ocorre entre as estratégias de treinamento empregadas. O aprendizado em um RNA, com ou sem a presença de supervisão, ocorre após uma fase, geralmente longa, de exposição a estímulos (exemplos ou padrões). O aprendizado em RBC concentra-se no armazenamento dos exemplos, que serão usados apenas quando o sistema for solicitado (aprendizado preguiçoso ou *lazy learning*).

Contudo, após uma primeira inspeção, os dois paradigmas não apresentam uma distância demasiada em suas abordagens, sugerindo mesmo uma continuidade de funcionamento modificada pelo nível de representação. Quando, por exemplo, é projetada uma hibridização envolvendo uma otimização por um Algoritmo Genético e uma RNA, esta distância pode ser considerada maior, conceitualmente. Ainda que esta hibridização seja perfeitamente possível e eficiente, a inspiração não será totalmente biológica. O processo evolutivo é um fenômeno externo aos sistemas. Hibridizar RBC e RNAs, por sua vez, sugere o comportamento heterogêneo verificado biologicamente, em que existem camadas processando estímulos sensoriais interagindo com outras que constróem e interpretam um modelo da realidade.

Da observação de sistemas modulares e heterogêneos agindo com sucesso em conjunto no mundo real, origina-se a decisão de usar um sistema híbrido. Da distância menor entre RBC e RNAs provém a escolha dos dois paradigmas.

Assim sendo, a experimentação com um sistema híbrido pareceu ser a mais indicada em novas tentativas de implementação de sistemas inteligentes, como a que agora se desenvolve.

Essa opção pelo emprego de sistemas que não se restrinjam à aplicação de uma só abordagem típica, (Regras de Produção, Redes Neurais, Algoritmos Genéticos, entre outros) mas que incorporem estratégias mistas, vem apresentando uma grande taxa de sucesso entre as aplicações divulgadas ultimamente [Zeghib et al. 2001].

A pesquisa e a utilização prática de RNAs levou à disseminação desta técnica eminentemente interdisciplinar, com a conseqüente determinação de uma série de problemas cotidianos, em diferentes setores, adequados à abordagem conexionista. Porém, ainda hoje, a maior parte do projeto de uma solução baseada em RNAs possui uma base empírica, levando em conta experiências anteriores, em sua maior parte de um grupo restrito de profissionais. Algumas tentativas para o estabelecimento de uma teoria, ou pelo menos de regras para a configuração desses parâmetros têm sido empreendidas, na maioria dirigidas a subconjuntos definidos do problema geral [Hilario 1996][Reategui, Campbell 1994]. A própria área enquadra-se, portanto, no que se poderia chamar de aplicação com modelo fracamente definido. Entende-se aqui uma aplicação de modelo fracamente definido como sendo aquela em que o conhecimento que se necessita representar é incompleto, impreciso ou mesmo contraditório. A representação poderia ser realizada através das técnicas usuais (regras de produção, *scripts*), mas não se tem acesso a toda a informação necessária, ou essa informação possui um grau de complexidade e interdependência interna em seus elementos que inviabiliza uma forma explícita. Podem mesmo existir situações em que o conhecimento adquirido apresente contradições, seja por uma questão de aquisição adequada, seja por uma influência de contexto difícil de ser compreendida.

## 1.2 Objetivo

O objetivo deste trabalho é a proposta de um modelo híbrido em que possam ser aproveitadas as principais alternativas existentes para a fusão das duas abordagens envolvidas e que possa ser comparado com alternativas puramente conexionistas e puramente simbólicas.

Para tal, o trabalho compõe-se de três partes distintas:

uma proposta de **abordagem híbrida** interna a RBC;

a implementação de um **ambiente de desenvolvimento** de sistemas de RBC;

uma **aplicação** da metodologia RBC, híbrida ou não, ao problema de escolha e configuração de RNAs.

A abordagem híbrida proposta será detalhada no Capítulo 3. Os dois outros componentes do trabalho envolvem a implementação de um protótipo de sistema que use

RBC. Esse sistema foi inicialmente testado na resolução do próprio problema enfrentado por projetistas de RNAs quando da escolha de um modelo adequado a uma determinada tarefa.

Dessa forma, a pesquisa procura não somente propor novas técnicas, mas também gerar um ambiente de desenvolvimento que possa ser utilizado no futuro, em outros trabalhos de pesquisa e também como auxílio didático. O trabalho compreende também, como passos intermediários e intrínsecos, o estudo de definições mais naturais de casos e a determinação do perfil dos problemas mais adequados a este tipo de abordagem híbrida. As aplicações iniciais para as redes escolhidas relacionam-se a tarefas de reconhecimento de padrões em um ambiente real.

## 1.3 Características da Aplicação

Um das etapas deste projeto, conforme foi explicado na seção anterior, consiste em implementar um programa que oriente o usuário, leigo ou não, quando do desenvolvimento de um novo sistema conexionista. Dentro do escopo deste trabalho, um sistema conexionista significará uma RNA.

Fundamentalmente, uma RNA é um sistema composto por diversas unidades simples de processamento que, trabalhando interconectadas e em paralelo, formam um sistema que muda de estado conforme recebe um estímulo. Aqui, trata-se de redes artificiais para reforçar-se a distinção com sistemas do mesmo tipo que ocorrem naturalmente na fisiologia animal. Essas redes orgânicas, entre as quais inclui-se o sistema nervoso dos mamíferos de um modo geral, foram as inspiradoras da idéia por trás do paradigma das RNAs em computação. Note-se também que as redes aqui referidas são simulações em *software* dos sistemas anteriormente mencionados, sobretudo no que diz respeito ao paralelismo de seu funcionamento. Implementações em *hardware* estão fora do escopo deste trabalho.

Para tal, foi iniciada a construção de uma base de casos de modelos, bem sucedidos ou não, de RNAs. Esta base constitui ao mesmo tempo a peça mais importante para a implementação do vertente simbólica do sistema e a que envolve a maior dificuldade de execução. Optou-se pelo não estabelecimento de um modelo rígido para a definição de uma RNA, conforme é indicado pelo paradigma de RBC, ou seja, foi escolhido um conjunto mínimo de atributos necessários. Assim sendo, os casos de redes inicialmente serão descritos por um conjunto variado de atributos, determinados pelos usuários originais que os implementaram. A determinação da topologia e outras grandezas numéricas envolvidas no processo será realizada baseada em regras de pré-processamento, ferramenta usual dentro de RBC e disponível no ambiente desenvolvido ao longo deste trabalho. Porém, a construção

dessa base nesses termos puramente de instâncias, ou seja, fortemente dependente da diversidade de casos conseguida, é mais lenta. Assim sendo, a etapa de desenvolvimento do ambiente de RBC, a ser mostrada mais adiante no Capítulo 4, foi antecipada, seguindo a construção da base em paralelo.

De modo a possibilitar o aproveitamento de uma característica muito interessante do problema de configuração de uma RNA, escolheu-se determinar a medida do desempenho das redes através de sua implementação em um simulador. Essa característica abre a possibilidade de automatizar a etapa ( do ciclo de RBC) de reutilização de uma solução. A solução proposta pode chegar mesmo a prescindir da análise de um usuário, tendo seu sucesso ou não determinado por uma execução da RNA, contanto que os arquivos de dados para teste e validação estejam disponíveis.

Uma vez estabelecido que as redes produzidas serão testadas em um simulador, de modo a fornecer dados ao módulo de RBC que permita sua avaliação, uma primeira idéia foi adotar o padrão empregado pelo simulador SNNS (Stuttgart Neural Network Simulator). O SNNS é bastante empregado atualmente, no ambiente acadêmico, para testes de modelos neurais [Zell et al. 1995]. Ele possui características extremamente interessantes para o projeto: seu código-fonte está disponível para recompilação, é gratuito, confiável, largamente utilizado em diversos países e grupos- de pesquisa, e possui uma gama bastante grande de modelos implementados. Além disso, já foi desenvolvido em nosso laboratório um simulador de RNAs, o Kipu [Vargas et al. 1997], projetado para ser executado no ambiente Windows e utiliza o mesmo formato de arquivo que o SNNS na descrição das redes. Além dessas opções, que servirão de ferramentas para a etapa de reutilização (reavaliação) das soluções propostas pelo sistema híbrido, uma terceira alternativa foi abordada: a interação direta com o simulador NeuroSolution, da Neuron Data Inc. [Ellis 1997]. Este simulador, apesar de não ser um software aberto e de distribuição gratuita, apresenta a vantagem de possuir um interfaceamento direto, tanto em Windows como em Unix, com qualquer ambiente RBC disponível, uma vez que pode ser executado tanto através de bibliotecas compartilhadas como por ativação direta (OLE, *Object Linking Embedding*, no caso do Windows). Isto viabilizou a construção de um programa único, sem a inserção de etapas manuais, facilitando sobremaneira a quantidade de testes realizados como diversas variações em torno de um mesmo modelo de RNA.

Dessa forma, restava a escolha de um ambiente de desenvolvimento no qual fosse implementada a metodologia de RBC. Existem ambientes de desenvolvimento disponíveis, e os principais foram objeto de atenção. Na verdade, a idéia da criação de ambientes (*shells*) de desenvolvimento em RBC existe desde o surgimento da área, como acontece com várias outras

abordagens no âmbito de Inteligência Artificial, como Algoritmos Genéticos e as prósprias RNAs. Atualmente, porém, a grande maioria dos programas existentes é de propriedade de empresas desenvolvedoras de *software*, trabalhando como consultoras. Alguns desses programas são referenciados no próximo capítulo.

Os ambientes comerciais demonstram tanto o potencial de interesse pela abordagem de RBC quanto seu possível grau de sofisticação. São, contudo, em sua maioria, produtos de preço elevado, estrutura interna pouco transparente e que não permitem alterações em seus algoritmos inteligentes básicos.

Assim, o caminho julgado mais adequado ao bom andamento da pesquisa foi o de desenvolver um ambiente próprio de RBC. Esse alternativa apresenta também a vantagem de proporcionar uma compreensão melhor das características da metodologia de RBC, em virtude do contato com as questões levantadas à medida que o programa foi sendo desenvolvido.

O ambiente de desenvolvimento recebeu o nome de RaBeCa ( de Raciocínio Baseado em Casos ) e, apesar de ter sido imaginado com a característica intrínseca de ser uma ferramenta de teste para hibridizações em RBC, pode ser considerado totalmente concluído ao final desta tese, pois encontra-se no momento em condições de executar o ciclo básico de RBC indicado na Figura 2.1, executar algumas conjuntos de regras e estabelecer generalizações. Suas principais características e potenciais serão mostrados no Capítulo 4, tendo sido também apresentados à comunidade acadêmica em exposições orais em Congressos nacionais e internacionais. [Sovat, Carvalho 1999] [Sovat, Carvalho 2001a][Sovat et al. 2001].

## 1.4 Organização

O segundo capítulo traz uma explanação geral do que entende-se por Raciocínio Baseado em Casos. Esta explanação envolve o enunciado do paradigma fundamental de RBC e uma descrição de suas origens. Em seguida é apresentado o ciclo básico empregado atualmente quando da aplicação de RBC a um problema. Cada etapa desse ciclo é analisada individualmente. Uma breve descrição de alguns ambientes de desenvolvimento de sistemas de RBC incluída também foi incluída.

No terceiro capítulo, é apresentada e discutida uma proposta de hibridização entre a vertente simbólica representada por RBC e a abordagem conexionista das RNAs. Abordagens híbridas anteriores são analisadas brevemente, no tocante aos pontos de hibridização e possibilidades em aberto. As abordagens propostas nesta tese são então descritas, através da consideração de possíveis vantagens e desvantagens em relação a estratégias não híbridas e da razão pela qual tais hibridizações foram pensadas no tratamento do domínio focalizado neste

trabalho. Os detalhes da implementação são em seguida introduzidos, bem como os problemas enfrentados e soluções encontradas.

O ambiente de RBC desenvolvido para a implementação dos experimentos da tese é descrito no quarto capítulo, de modo que suas possibilidades e restrições, que inevitavelmente moldaram este desenvolvimento, sejam compreendidas e o tipo de representação do problema fique clara.

O processo de montagem da base de casos, ou seja, as instâncias de RNAs empregadas na tese, é explicado no capítulo 5. Essa explicação procura expressar o porquê da escolha dos atributos de caso, da sua organização na memória e da importância dos índices escolhidos. Após uma breve descrição do que se entende por um modelo conexionista, são descritos os principais parâmetros envolvidos em sua definição, isto é, tudo aquilo que realmente é passível de atenção por parte de um projetista e que influi no comportamento final do modelo. Basicamente, estes parâmetros são responsáveis pela maior parte da complexidade do espaço de soluções envolvido. Além disso, é mostrado como os recursos disponíveis no ambiente RBC foram usados para a manipulação dessa base: regras e generalizações. Finalmente um breve exemplo dos casos é apresentado.

O capítulo 6 comenta processos empregados para avaliação tanto da hibridização (inserção de uma RNA no ciclo) como do sistema final (escolha de um modelo de rede). As peculiaridades e possibilidades de métodos e critérios são abordadas, bem como uma metodologia para teste do sistema híbrido.

Um estudo de caso é apresentado no Capítulo 7, com uma subseqüente apresentação e discussão dos pontos fortes e fracos da abordagem utilizada.

Finalmente, as principais observações, conclusões, contribuições e pontos de desenvolvimento futuros são tratados no Capítulo 8.

Os apêndices incluem exemplos de casos e a descrição da linguagem CASL, usada no ambiente RBC. O código dos programas desenvolvidos também está disponível em meio magnético e na página do LABIC na Internet (www.icmc.sc.usp.br/labic).

# Capítulo 2

# Raciocínio Baseado em Casos

Esse termo é empregado para definir um das abordagens incluídas em Inteligência Artificial e que emprega técnicas de aprendizado por instâncias. Talvez seja mais adequado classificar-se RBC como sendo uma metodologia do que como uma técnica propriamente dita. Diferentemente de outras abordagens, já mais comuns na área de Inteligência Artificial, como o uso de lógica *fuzzy* ou RNAs, o aprendizado de um sistema de RBC dá maior ênfase à organização da memória e à escolha de um conteúdo para os dados que seja o mais significativo possível. Ele é também associado a abordagens ditas preguiçosas (*lazy learning*), uma vez que o processo de raciocínio em si acontece em uma única etapa e somente após o sistema ser exposto ao objetivo desejado. A resolução de problemas através de RBC evoluiu a partir dos sistemas baseados em regras de produção propondo uma estratégia mais ambiciosa. Enquanto aqueles sistemas tentam atingir uma solução apenas através do encadeamento de regras pré-definidas a respeito de um determinado tema, interagindo com o usuário através de perguntas e respostas, métodos de RBC operam de uma forma diferente e menos focalizada em um modelo.

## 2.1 Descrição da Metodologia

A idéia principal que sustenta os métodos de RBC é a de resolver novos problemas através da adaptação de soluções que funcionaram anteriormente em problemas similares [Kolodner 1993]. A área floresceu a partir de propostas e pesquisas realizadas por Roger Schank sobre memória dinâmica na Universidade de Yale, na década de 80, e tem sua plausibilidade baseada no raciocínio analógico e em teorias clássicas de formação de conceito [Schank 1982].

Depois de fornecida uma especificação do problema, um programa de RBC irá procurar em sua coleção de casos por um em que a definição do problema se assemelhe o melhor possível à entrada fornecida. Na melhor das hipóteses, será encontrado um caso cujo

problema se iguale àquele que foi proposto. Essa possibilidade aumenta, como é de se esperar, na razão direta da quantidade de casos que já foram apresentados ao sistema, ou seja, diferentemente dos sistemas de regras, programas de RBC aprendem pela experiência de uso, característica encontrada, mencione-se de passagem, em alguns modelos de Redes Neurais, por exemplo os modelos ART [Carpenter, Grossberg 1987]. Se uma solução idêntica não for achada, o usuário (ou mesmo o próprio programa) irá trabalhar com a melhor aproximação fornecida pelo método, ajustando a solução nas partes em que esta não atender aos requisitos, isto é, procedendo a uma adaptação do caso, a qual, por sua vez, também gera um caso novo que pode ser incorporado à base. Desta forma, se o mesmo problema for futuramente apresentado ao sistema, toda essa etapa de ajuste pode ser evitada e o resultado anterior será apresentado. Para que tal mecanismo funcione, um sistema que empregue RBC é composto fundamentalmente da referida coleção de dados, convenientemente estruturados e que sejam relevantes (base de casos), de uma estratégia de representação e armazenamento desses casos, de regras para adaptação de casos e de um mecanismo que permita medir-se a similaridade do problema proposto em relação àqueles guardados na memória. Podem ser ainda incluídas facilidades de consulta ao usuário quanto à adequação da solução proposta pelo sistema, quanto à necessidade de adaptações na mesma e quanto à validade ou não da inclusão do novo caso adaptado à base de casos. Em geral, essas facilidades consistem de perguntas diretas ao usuário através da interface do sistema, mas uma das frentes da pesquisa atual é justamente a criação de métodos eficientes para a automação dessas tarefas.

## 2.2 Ferramentas Existentes

Um pequeno e nem sempre atualizado conjunto de ferramentas encontra-se disponível gratuitamente, inclusive seu código-fonte. Neste grupo destacam-se, além do CASPIAN [Pegler, Price 1996], da Universidade de Wales, Grã-Bretanha, dois outros produtos também ligados ao meio acadêmico: CASUEL [Casuel 1994], da Universidade de Kaiserslautern, Alemanha, e PROTOS [Bareiss 1989] da Universidade do Texas, EUA.

O ambiente CASPIAN, que foi utilizado como base para uma linguagem descritora, como será explicado adiante, é um ambiente gratuito originalmente criado em DOS e que pode ser executado em uma janela do Windows ou em ambiente UNIX, porém sem interface gráfica. Contudo, seu mecanismo interno é sofisticado o suficiente para que já tenha sido empregado no desenvolvimento de aplicações práticas industriais [Price et al. 1997]. Ele admite multi-indexação e regras de pré-processamento e de adaptação.

CASUEL, por sua vez, não é um ambiente de desenvolvimento, é uma linguagem orientada a objetos, baseada em frames, bastante abrangente e destinada a representar e armazenar modelos descritivos e bibliotecas de casos em arquivos texto. Esta linguagem possui hierarquia de casos, herança de atributos, mecanismo para definição de similaridade, sintaxe para regras de adaptação e regras de pré-processamento. Foi criada com o objetivo de tornar-se o padrão europeu de representação de casos, existindo diversas aplicações acadêmicas desenvolvidas com sua utilização, porém sem um ambiente único de programação e teste. Cada base é desenvolvida e gerenciada por aplicativos específicos.

O terceiro *software* citado, PROTOS, é um sistema que aprende a partir de instâncias, sendo, portanto, um exemplo de programa de RBC, já que pode-se considerar esta área como uma subdivisão da de *Instance-Based Learning*, IBL, uma vez que aprende a partir de exemplos [Aha et al. 1991]. Protos não dispõe também de um ambiente integrado para desenvolvimento. Foi escrito primeiramente em Prolog e reescrito em Common LISP posteriormente. Apesar de necessitar de um interpretador LISP, é um dos primeiros experimentos com a metodologia RBC e já foi empregado em aplicações reais, constituindo-se em uma referência importante.

Finalmente, no que diz respeito a *software* comerciais, em grande parte dos casos, ele é direcionado a um tipo específico de aplicação, geralmente a aplicação que é a área principal de atuação do distribuidor. Dentre aqueles que são vendidos fundamentalmente como ambientes genéricos, e a que foi possível ter acesso, ao menos na forma de versões para demonstração, devem ser mencionados: o ESTEEM, da Esteem Software Inc. [Esteem 2001], na Califórnia e o CBR-Works, da TecInno GmbH, Alemanha. [Watson 1997]

O primeiro é um ambiente de desenvolvimento completo, projetado para rodar também em MS-Windows. Ele suporta o desenvolvimento de aplicações que acessem múltiplas bases de casos ao mesmo tempo e atributos que sejam em si outros casos aninhados. Fornece também vários métodos de definição de similaridade, tais como número de atributos, cálculo de pesos de atributos e inferência sobre as características de um caso. Possui editor de casos e de regras de adaptação e pré-processamento. Esse ambiente destaca-se também por possuir dois mecanismos de indexação: um indutivo, utilizando o algoritmo ID3, apresentado por J. R. Quinlan [Quinlan 1986], para geração dos pesos de atributos e outro alternativo, baseado em um método de gradiente descendente.

Outra alternativa comercial, o CBR-Works foi desenvolvido de modo a ser independente de plataforma. Essa característica vem do fato de ter sido criada originalmente em SmallTalk. Uma vez que pode ser interpretada, dispôs de versões também para

Windows95/98. É um ambiente bastante sofisticado no qual pode-se desenvolver uma aplicação completa de RBC, dividido em facilidades específicas para diagnóstico, recuperação de informações em textos (classificação) e suporte a usuários (*help desk*). Esse *software* é na verdade uma versão comercial de um conjunto de ferramentas e diretrizes criadas também na Universidade de Kaiserslautern, sob o projeto INRECA [Johnston et al. 1996], utilizando CASUEL como linguagem descritora. Possui editor de atributos, editores de regras, editor de critérios de similaridade, suporte para processamento distribuído, facilidades de acesso a bancos de dados e para integração a aplicações já existentes.

## 2.3 Ciclo Básico de Funcionamento

O ciclo de funcionamento geralmente adotado para um sistema de RBC envolve quatro etapas (4 "RE's") : recuperação, reutilização, revisão e retenção de casos [Aamodt, Plaza 1994], conforme mostrado na Figura 2.1.

Após o usuário apresentar seu problema em uma forma reconhecida pelo sistema de RBC, o sistema escolhe um caso que tenha sido julgado o mais adequado, ou que mais se aproxime ao apresentado, através da aplicação de sua ferramenta de similaridade à base de casos existente. Essa primeira fase, recuperação, pode ser precedida de processos automáticos de pré-processamento, de modo a aumentar suas chances de êxito. Nesses processos, são acionadas regras de produção que podem manipular os atributos originais, não exatamente modelando a área de aplicação, mas sobretudo introduzindo o máximo possível de senso comum e conhecimento do mundo.

Em seguida, o caso recuperado da memória é mostrado ao usuário para que ele possa decidir sobre a possibilidade de utilização dessa solução anterior em seu problema atual. Esta é a segunda fase, reutilização. Neste ponto pode-se em geral aceitar-se a solução, optar-se pela busca de um segundo caso na memória ou seguir para a próxima fase. Caso a solução proposta não seja totalmente satisfatória, entra-se na fase de revisão, na qual, mais uma vez, o usuário, ou processos de adaptação estabelecidos *a priori* por regras do mesmo tipo que as de pré-processamento, são chamados a transformar a solução, visando a aproximá-la ainda mais do problema proposto. Essas duas fases geralmente compõem um subciclo, que se repete até que a solução seja considerada adequada. Ao término desse subprocesso iterativo (reutilização-revisão), o sistema deve decidir, novamente recorrendo à intervenção do usuário ou a critérios pré-determinados, se o novo caso, obtido da união do problema apresentado com a solução proposta, deve ou não ser adicionado à base de casos. Esta última fase, retenção, constitui efetivamente a etapa de aprendizado do sistema.

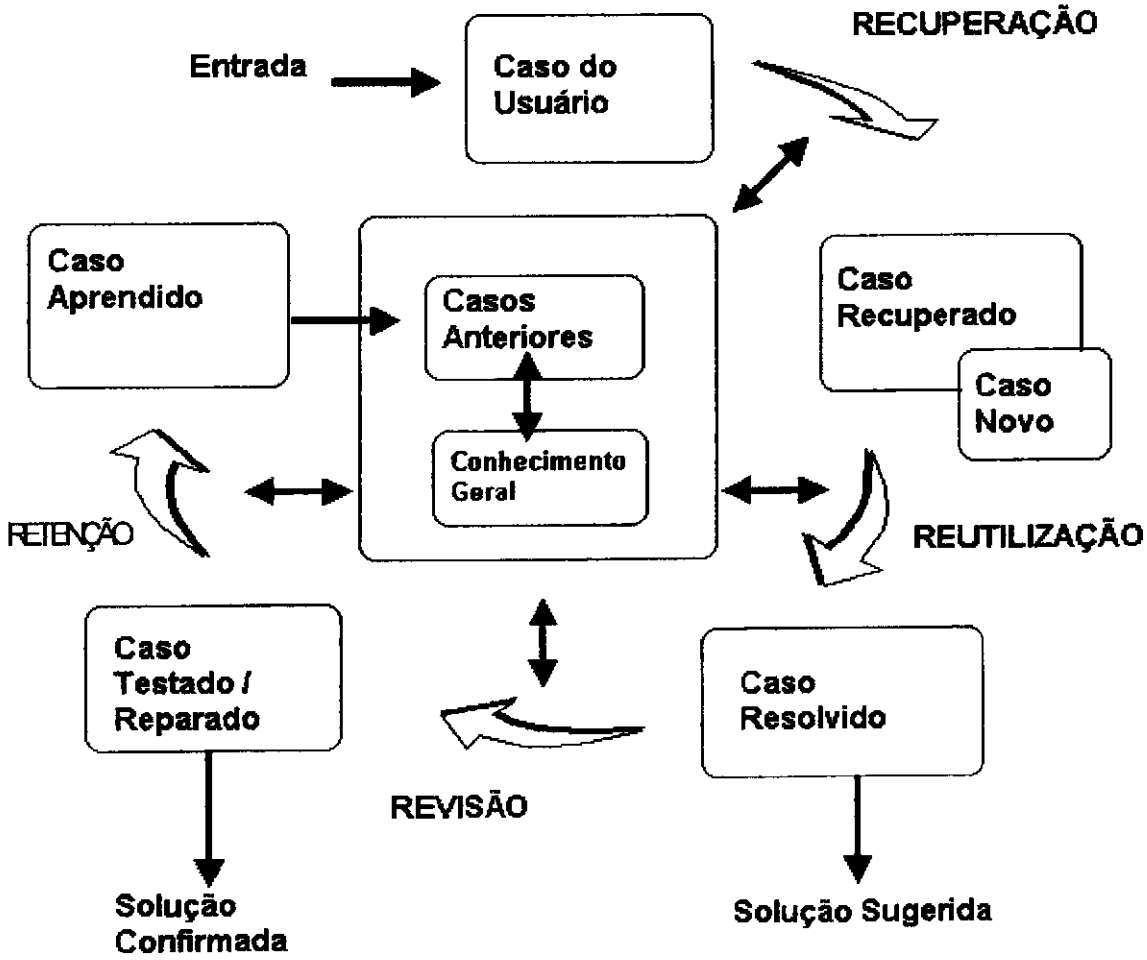


Figura 2.1 – O ciclo básico de funcionamento de RBC.

É importante ressaltar mais uma vez que o ciclo apresentado pretende contribuir não com uma nova tecnologia, mas sim com um método consistente de aprendizado e raciocínio [Althoff, Wess 1992] [Watson 1998]. Dentro dessa metodologia, a adequada escolha e combinação de técnicas disponíveis no campo da Inteligência Artificial e em áreas afins permite a construção de sistemas com características bastante desejáveis: são adaptáveis a atividades humanas onde não se dispõe de um modelo formal ou mesmo aproximado do conhecimento; minimizam, por essa razão, a etapa de aquisição de conhecimento e seguem um processo de raciocínio que talvez seja o mais natural para o ser humano, que é a analogia entre diferentes situações de modo a construir um modelo da situação real corrente. Esse tipo de raciocínio tem sido empregado com sucesso sobretudo em atividades que envolvam qualquer tipo de diagnóstico (medicina, detecção de falhas) [Gierl, Stengel-Rutkowski 1994] [Weiner et al. 1995], planejamento de um modo geral [Carbonnel et al. 1991], processos que exijam projeto (*design*) [Maher, Pu 1997] e em classificação de padrões [Grimnes, Aamodt 1996]. A área de pesquisa em si é relativamente nova, pois apesar de seus fundamentos terem surgido há quase vinte anos, as referências que deram nome e especificidade ao RBC datam do fim da década de 80 e ao início da década de 90 [Riesbeck, Schank 1989][Watson, Marir 1994]

[Kolodner 1993]. Seu aspecto mais abstrato e o fato de ser direcionada a aplicações provenientes do mundo real contribuem para uma melhor aceitação e disseminação junto à indústria e a alguns setores de serviços, já tendo sido relatados vários desenvolvimentos de sucesso, com a conseqüente abertura para o emprego de técnicas clássicas de Inteligência Artificial [Acorn, Walden 1992] [Heider 1996].

## 2.4 Adaptação de Casos

De modo a tornar mais claro o procedimento de adaptação de um caso, será feito a seguir um pequeno resumo dos modos de operação dessa etapa dentro da metodologia de RBC.

Apesar de ser aquela de maior interação com o usuário, e talvez justamente por isso, a fase de adaptação no ciclo RBC é aquela que ainda necessita de maior pesquisa para determinação de um conjunto de métodos e diretrizes em seu uso. Em virtude de ser a fase em que o usuário transforma o caso resgatado em um caso-solução para seu problema, ela concentra as peculiaridades do domínio escolhido e o conhecimento do especialista que montou a base de casos. Sua automação, portanto, não é uma tarefa trivial. A maior parte dos processos de adaptação utilizados atualmente funde-se à fase de avaliação do caso transformado em uma grande sucessão de diálogos de consulta ao usuário. Parte dessa tarefa, porém, pode ser implementada através de regras de produção, chamadas de regras de adaptação, ou regras de reparo.

Essas regras executam um subconjunto de operações básicas necessárias à adaptação de atributos de um caso. Essas operações foram propostas pelos pesquisadores da área de RBC desde o início de seu estabelecimento [Schank 1982]. Basicamente, elas dividem-se em dois grupos: **substituições** e **transformações**. Além desses dois tipos, incluem-se métodos específicos a determinados ambientes e uma técnica, conhecida como *Derivational Replay* [Carbonell et al. 1991], que consiste em utilizar a própria idéia de RBC para tentar enquadrar o processo de adaptação atual em uma adaptação bem sucedida anterior. Essas duas últimas técnicas não são executadas a partir de regras, não sendo abordadas nesta tese, mas parte das duas primeiras foram incluídas e serão indicadas a seguir sempre que um exemplo de seu tipo ocorrer.

### 2.4.1 Métodos de Substituição

Fundamentalmente, procuram alterar um valor de um atributo do caso, ou variar uma grandeza numérica, ou mesmo de um grupo de atributos:

Reinstanciação: troca um valor do caso resgatado por outro em todos os pontos em que ele ocorrer, de forma a condicioná-lo dentro de requisitos do caso proposto. No domínio de RNAs, poderia ser, por exemplo, a escolha de um modelo diferente para efetuar uma tarefa de classificação realizada anteriormente. Neste trabalho, como será mostrado no capítulo 4, consegue-se este efeito através das regras que usem **<change ... to ...>** em seu conseqüente. Essa substituição é mais fácil de ser generalizada e automatizada. Ocorre mais em regras locais, nas quais sabe-se que se o caso for escolhido, é necessário mudar algum atributo em particular.

Ajuste de Parâmetros: altera valores numéricos do caso resgatado, geralmente segundo uma heurística adequada à obtenção de uma solução com maior possibilidade de êxito. Por exemplo, altera a topologia de uma rede bem sucedida anteriormente para adequar-se a um novo tipo de padrões. Dentro do desenvolvimento desta tese, este método é implementado por regras que usam **<evaluate ... to ...>**. Geralmente é uma substituição baseada em uma heurística, ou conhecimento específico do construtor da base a respeito do domínio. Ocorre mais freqüentemente em regras de pré-processamento e regras locais, caracterizando uma maior inserção de conhecimento na base de casos pelo especialista.

Busca Local: procura alternativas em uma estrutura de memória auxiliar a fim de encontrar substitutos para valores não adequados. Não foi necessário utilizar este tipo de substituição nesta tese, mas pode-se executar este método através das abstrações. Se um determinado valor está na lista de abstrações de um atributo, ele pode ser substituído por regras **<change ... extract ... from ...>**. Essa tipo de substituição é o mais comum em regras globais, pois torna-se mais flexível para todos os casos.

Memória Auxiliar: implementa explicações em uma estrutura em separado para fornecer ao usuário maiores informações que possa encontrar sobre um atributo.

Busca Especializada: executa a mesma procura por explicações mas permite que essa busca seja orientada por uma heurística própria, de modo a encontrar melhores detalhes ou fazer inferências.

Substituição de Caso: busca em outros casos, que não aquele que foi resgatado em primeiro lugar, sugestões para novos valores.

### 2.4.2 Métodos de Transformação

As substituições vistas no item anterior só podem ocorrer quando o caso resgatado possui um atributo a ser adaptado. As transformações ocorrem quando é necessário criar ou remover um atributo de um caso.

Basicamente, existem dois tipos de transformação:

Transformação baseada em senso comum: apóia-se em uma heurística, de modo que usa o conhecimento a respeito da importância relativa de um atributo do caso. A partir dessa heurística, esse atributo é levado em consideração com maior ou menor influência, podendo ser mesmo eliminado. Em geral, essa informação é extraída do conhecimento do mundo real. Apesar de não ter sido necessária na execução deste trabalho, ela pode suprimir um atributo de caso através de regras que usem **<change weight of ... to ...>** alterando seu valor para zero. Quando essa supressão ocorre em regras de pré-processamento, os casos podem sofrer um ajuste de acordo com o senso comum, conforme foi descrito. Não há a possibilidade, porém, de introduzir-se um novo atributo de caso que não tenha nenhuma relação com os existentes.

Reparo guiado por modelo: lança mão de relações causais dentro do domínio explorado. Atributos são incluídos ou removidos de acordo com o conhecimento prévio de sua ocorrência ou não a partir de outros já existentes. Esse tipo de alteração é constituído por um conjunto de heurísticas, mas que são criadas especialmente para a área dos casos. O mesmo tipo de mecanismo anterior é executado no Rabeca, desta vez durante a aplicação de regras de reparo globais e locais.

## 2.5 Considerações Finais

Neste capítulo, foi apresentada a metodologia de RBC. Não se pretende que a terminologia empregada, bem como a definição do ciclo mostrado, constituam-se em um padrão, mas elas seguem as convenções mais comuns, encontradas na bibliografia. A maior ênfase nos processos de adaptação deve-se à importância da compreensão desta etapa para o prosseguimento da explanação da abordagem híbrida apresentada nesta tese. Esta abordagem será mostrada no capítulo a seguir.

# Capítulo 3

# Uma Abordagem Híbrida

Atualmente, abordagens híbridas ao problema de configuração de RNAs começam a ser propostas, envolvendo regras de produção ou tipos de representação do conhecimento mais tradicionais (planos, *scripts* e mesmo bancos de dados relacionais) [Surma, Vanhoof 1995] [Holt, Benwell 1999]. Outras tentativas limitam-se a aplicar um módulo conexionista em uma das etapas do ciclo de RBC [Aamodt, Plaza 1994]. A contribuição que se propõe nesta tese é a avaliação da possibilidade de reunir duas metodologias de uma forma adequada, uma complementando a outra de acordo com as características do problema : RBC, mais nova, que vem tomando corpo nos últimos dez anos, e RNAs, uma área de pesquisa que, além de lhe ser próxima, apresenta problemas de projeto adequados à primeira.

## 3.1 Abordagens Anteriores

A introdução de uma RNA em um ciclo RBC já foi proposta anteriormente por alguns grupos de pesquisa [Shin, Park 2000], [Corchado, Lees 2000], [Malek 2000], [Fabiunke et al. 1997], [Domeshek 1993], [Petridis, Paraschidis 1993], [Sase et al. 1993], [Wendel 1993], [Becker, Jazayeri 1989], [Thrift 1989]. Esta idéia deve ser vista não somente como um tópico de pesquisa, mas sobretudo como uma tendência natural na metodologia de RBC: a integração de RBC com outras técnicas inteligentes.

De acordo com estes trabalhos anteriores, a integração pode ser atingida tanto através da divisão de tarefas entre o sistema de RBC e a RNA, como pelo projeto de uma arquitetura inteligente que combine as características de RNAs e RBC. No que diz respeito ao primeiro tipo de abordagem, existem, de acordo com Reategui and Campbell [Reategui, Campbell 1994], quatro alternativas que englobam e generalizam os diferentes paradigmas encontrados:

**Controle Central**, em que tanto o RBC quanto a RNA são controlados por um dispositivo central.

**Controle Distribuído**, em que o controle é dividido entre as duas técnicas.

**Rede Neural Dominante**, no qual o controle favorece a RNA.

**Raciocínio Baseado em Casos Dominante**, em que o controle favorece o sistema de RBC.

O trabalho desenvolvido nesta tese enquadra-se na última destas alternativas. O sistema final planejado consiste fundamentalmente de um sistema híbrido no qual o ciclo RBC determina o procedimento geral de raciocínio. Os módulos conexionistas que são introduzidos funcionam como ferramentas auxiliares.

Porém, a mais completa classificação de sistemas híbridos envolvendo conexionismo e simbolismo (sistemas neuro-simbólicos) foi realizada por Hilario em [Hilario 1997]. Basicamente, sua taxonomia segue a estrutura representada na Figura 3.1.

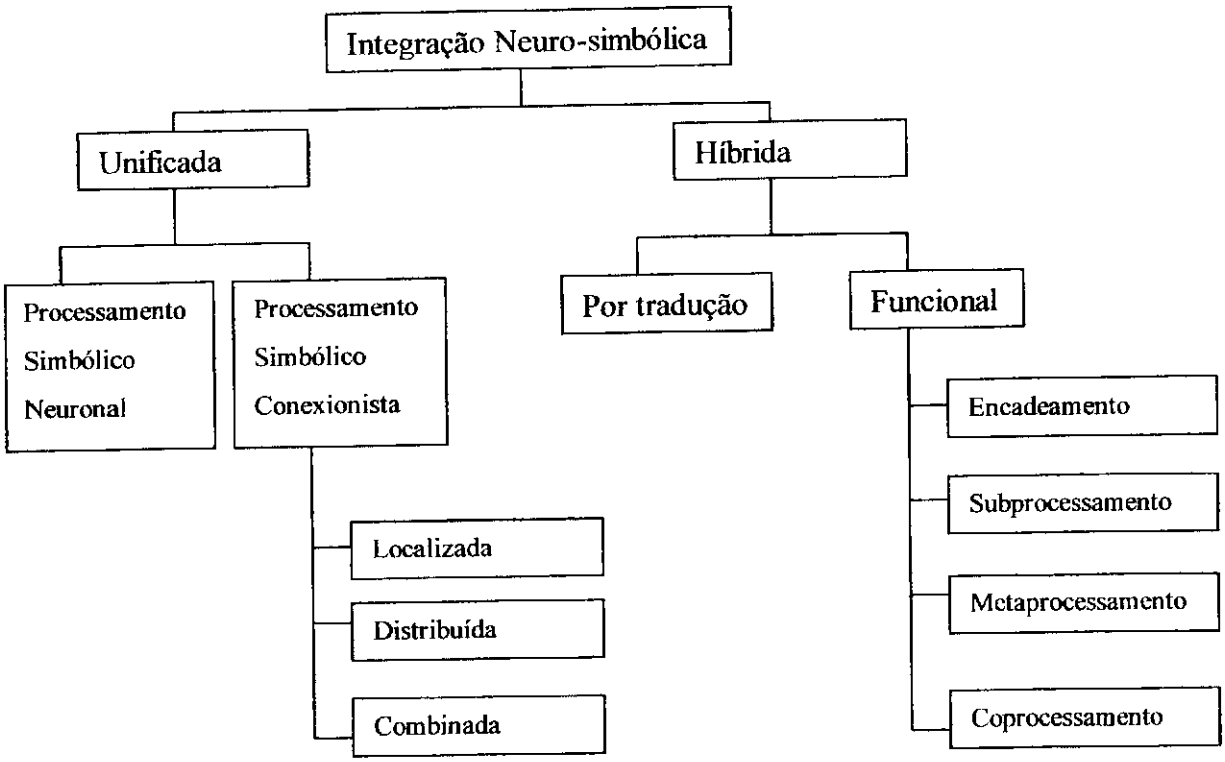


Figura 3.1 - Classificação de Sistemas Neuro-simbólicos Integrados

Antes de situar a abordagem proposta por esta tese na estrutura mostrada no diagrama anterior, é necessário explicar resumidamente cada um dos tipos de integração mencionados.

Inicialmente, existem dois tipos de integração possíveis, que formam a primeira grande subdivisão da estrutura:

**Integração Unificada**, que busca obter as características e possibilidades simbólicas e conexionistas lançando mão apenas de RNAs.

**Integração Híbrida**, que procura combinar RNAs com modelos simbólicos, como árvores de decisão e sistemas especialistas.

Dentro da Integração Unificada, por sua vez, foram estabelecidos dois tipos de processamento para que a integração ocorra:

**Processamento Simbólico Neuronal**, em que a diretriz básica é a plausibilidade biológica. O neurônio real, em suas diversas variedades, norteia este tipo de processamento que, normalmente, está associado a tentativas de modelagem e interpretação do funcionamento das funções superiores do cérebro.

**Processamento Simbólico Conexionista,** no qual, ao contrário da abordagem anterior, redes explicitamente artificiais são sintetizadas a fim de reproduzir um comportamento ou realizar uma função desejada. Esta abordagem parte do princípio que funções simbólicas complexas emergem diretamente de processos e estruturas neurais. Para tal, três tipos de arquitetura são encontrados em implementações:

Arquitetura Local, que mapeia unidades individuais da RNA em estruturas simbólicas específicas, ou seja, cada nó da rede representa um conceito ou um grupo de conceitos existente no conjunto de símbolos.

Arquitetura Distribuída, que surgiu em virtude de problemas de explosão no processamento baseado na abordagem local. Nesta arquitetura conceitos elementares emergem da combinação de unidades de processamento (nós). Cada item simbólico, por exemplo, um fato ou uma regra, é uma conseqüência da interação de diversas unidades.

Arquitetura Combinada, que procura reunir o poder de processamento local com a eficiência do processamento distribuído.

Por sua vez, a integração híbrida parte da premissa que apenas uma cooperação entre as características de cada mundo, ou seja, uma sinergia entre o simbolismo e o conexionismo, poderá realmente atingir uma capacidade computacional e cognitiva completa. Este tipo de integração apresenta duas vertentes principais :

**Integração por tradução**, ou por transformação é um meio-termo entre as abordagens unificadas e as demais híbridas. Também baseia-se apenas em RNAs como processadores, mas admite que o início ou o fim do processamento seja uma estrutura simbólica. Geralmente procura transformar essas estruturas em RNAs, de forma que estas não sejam manipuladas como tais. Por exemplo, regras de produção não são processadas por um motor de inferência, mas são utilizadas como fonte de conhecimento para a síntese de uma RNA que a expresse. É importante notar que nesta abordagem fica implícito que, apesar de assumir que sistemas puramente conexionistas podem expressar características simbólicas, a interação entre diversos

sistemas, ou entre eles e o usuário, exige uma transformação bidirecional entre as RNAs e representações simbólicas de alto nível.

**Integração Híbrida**, que incorpora componentes simbólicos e conexionistas completos. Além de possuir RNAs, compreende também estruturas simbólicas e seus processadores usuais (*parsers*, interpretadores de regras). Ao contrário da abordagem anterior, esta reforça a importância de obter-se uma interação real entre seus componentes. De uma forma geral, quando mencionam-se sistemas híbridos, está envolvida uma hibridização funcional. O grau da interação (frouxa ou restrita) é um indicativo do estabelecimento dessa hibridização, mas como essa gradação pode às vezes ser um pouco subjetiva, torna-se preferível indicá-la de uma forma qualitativa, expressando o modo pelo qual os componentes se relacionam. Supondo-se sempre ao menos um processador de cada tipo, quatro esquemas podem ser identificados, conforme também mostrado na Figura 3.2:

Processamento encadeado, em que um dos módulos é o processador principal enquanto o outro executa tarefas de pré-processamento ou pós-processamento.

Subprocessamento, no qual um dos módulos está embutido no outro e subordinado a ele. Existe apenas um módulo externo que atua como o solucionador principal, interagindo com o problema.

Metaprocessamento, que estabelece um módulo como solucionador de problemas em um nível básico, enquanto o outro participa em uma instância superior, melhorando o desempenho do primeiro ou supervisionando o processo. Ambos os módulos interagem diretamente com o ambiente.

Coprocessamento, em que ambos os processadores são parceiros de mesmo nível na solução do problema cada um interagindo diretamente com o ambiente e trocando informações entre si. Podem também competir internamente sob a supervisão de um metaprocessador ou dividir tarefas específicas.

Baseando-se na taxonomia resumida apresentada, a arquitetura proposta pode ser definida como uma abordagem híbrida, pois combina redes neurais com modelos simbólicos. Esta combinação será explicada nas próximas seções deste capítulo. Porém, como uma apresentação inicial, existe uma RNA operando em paralelo com um algoritmo k-NN, ambos inseridos em um ciclo RBC. Este ciclo RBC por sua vez dispõe de um analisador léxico (*parser)* para o *script* descritor, de um mecanismo de generalizações e de um processador de regras. Em virtude dessas implementações, a proposta também é subclassificada como funcional, porque incorpora esses componentes simbólicos e conexionistas completos.

Finalmente, ela possui um modo de integração em que o coprocessamento é mais notável, já que nenhum dos dois componentes (RNA e k-NN) possui o controle total da recuperação, apesar de não existir competição. Cada um deles interage diretamente com o ambiente e recebe informações um do outro. Dentro desta classificação, porém, o sistema apresenta também uma característica de metaprocessamento, já que o ciclo RBC manipula os casos em uma forma mais básica, sendo aprimorado pelo par RNA / k-NN.

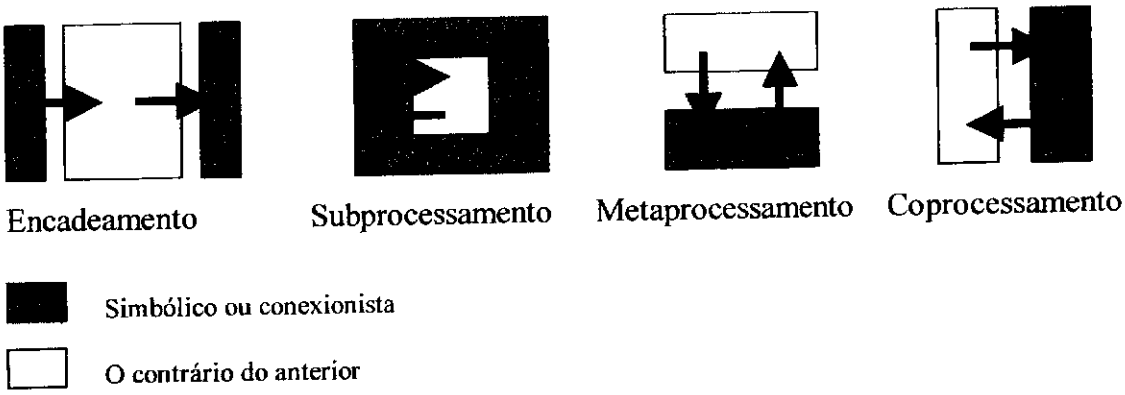


Figura 3.2 - Modos de integração neuro-simbólicos híbridos

## 3.2 Características do Problema

Algumas características especiais foram encontradas no perfil do problema de projetar um RNA, no que diz respeito à realização dessa tarefa levando-se em consideração casos anteriores. A seguir, essas características serão examinadas em seus pontos principais.

A primeira das particularidades é a existência de um número consideravelmente maior de atributos de solução do que de caso, o que seria de se esperar. Isto chama a atenção para a necessidade de uma orientação especial na busca pela solução ideal. Esse perfil mostrou-se provável após as primeiras etapas de aquisição de conhecimento e na escolha de exercícios que servirão de protótipos. Um problema a ser abordado por um RNA em geral consiste fundamentalmente de um conjunto de dados e uma questão levantada a respeito desses dados.

O usuário, um aluno, por exemplo, possui um conjunto de dados e deseja realizar uma tarefa de classificação ou regressão sobre ele. É quase toda informação relevante de que ele dispõe. Características físicas e específicas, envolvendo o ambiente computacional e o usuário, mostraram-se de pequena importância: o usuário escolhido como o alvo do sistema foi definido como sendo um leigo em sistemas conexionistas e o binômio máquina / sistema operacional tem sofrido uma padronização de desempenho em função do desenvolvimento da área em

geral. Na consulta desse usuário a um especialista humano, esse conjunto reduzido de informações seria o ponto de partida para sessões de projeto, usando-se um simulador ou não.

Em conseqüência, o problema concentra-se em corretamente escalonar os paradigmas mais bem sucedidos em cada tipo de tarefa e correlacionar os parâmetros estatísticos que descrevem os dados a fim de obter-se um estado inicial o mais próximo possível do objetivo. Essa tarefa, porém, não se mostra tratável por uma partição simples do espaço de soluções. Ao contrário, em cada paradigma escolhido, pequenas variações de parâmetros levam a grandes mudanças no desempenho da rede construída. Isto quer dizer que a implementação da escolha em um ciclo de RBC vai depender bastante da fase de adaptação.

Chegar a uma configuração inicial que possa ser considerada boa (por exemplo, cujo ganho de desempenho da rede não varie mais que 10% durante o processo ) pode ser conseguido através de um método de indexação direto, sem muita informação na estrutura da memória de casos, ou mesmo em memória *flat*. Porém, o refinamento irá exigir um percurso cuidadoso na alteração do caso selecionado da base. Esse percurso poderia ser realizado unicamente através das etapas de consulta ao usuário durante a adaptação, mas exigiria um nível de experiência que não é compatível com o perfil de usuário pré-determinado. Essa peculiaridade do problema não se constitui em um indicativo contrário à aplicação de uma solução baseada em casos. Em vez disso, apenas concentra seu uso naquela parte em que efetivamente a experiência do especialista faz-se notar: a escolha não do paradigma, mas do caminho a ser seguido durante o processo de ajuste da rede, ou seja, da seqüência de alterações de parâmetros usuais para obter-se a rede ótima. É interessante observar aqui que, enquanto o problema continua sendo essencialmente um problema de *design*, essa característica abre a possibilidade de investigação da conveniência em incluir algum tratamento temporal na solução, conforme ocorreria em uma solução na área de planejamento. Contudo, nesta primeira abordagem, esse tratamento não será incluído. As redes propostas como solução serão apresentadas como um todo, com todas as alterações de parâmetros ocorrendo simultaneamente.

A estratégia a ser usada para contornar o conflito entre a necessidade de adaptação pelo usuário e a pouca experiência dele com o domínio será a de aumentar o quantidade das regras de adaptação. A utilização de um conjunto de regras de produção dentro de uma base de casos, ainda que possa vir a ser considerada como uma fuga da idéia fundamental de RBC, é prática corrente em todos os sistemas baseados em casos que têm sido implementados sobre domínios do mundo real [Reisbeck, Schank 1989][Watson 1997]. Isso deve-se ao sucesso desse tipo de construção de bases, em que fica claro que, apesar do conhecimento

especializado principal ainda provir dos casos, existe a necessidade de um maior conhecimento do mundo, como em geral acontece com os demais tipos de sistemas inteligentes. A concepção do ambiente de desenvolvimento de Raciocínio Baseado em Casos desenvolvido durante este trabalho (RaBeCa) prevê essa possibilidade na forma de regras de transformação apenas, mas esse tipo de manipulação é suficiente para o tratamento em questão, uma vez que as tarefas de adaptação serão justamente transformações ( criação, alteração e remoção ) de parâmetros.

## 3.3 Abordagem Proposta

Resumidamente, procurou-se enfrentar dois problemas comuns em RBC. O primeiro é o excesso de rigidez das bases de casos [Leake et al. 1997]. Este chamado excesso de rigidez é decorrente do processo de indexação empregado.

Conforme foi visto no capítulo anterior, o mecanismo indexador é responsável por caracterizar um caso corretamente e escolhê-lo para que seja comparado com o caso apresentado pelo usuário. Esta escolha depende da organização da memória e dos índices escolhidos. Em uma memória hierárquica, os atributos valorados pelo usuário são usados para orientar gradativamente a busca na base, auxiliados pelos índices pré-estabelecidos. O processo de busca é guiado, portanto é mais eficiente. Em uma memória *flat*, os índices possuem um papel mais decisivo, eles constituem a única indicação de prioridade. Os pesos serão usados apenas na operação de casamento (*matching*) entre os casos. Os índices são também elementos de restrição da busca de forma a permitir a manipulação de bases de grande porte. O processo de indexação é, evidentemente, decisivo para uma apresentação adequada de um conjunto de casos pertinentes. Em ambos os casos de organização de memória, porém, há uma introdução de conhecimento na base. Uma escolha de hierarquias e índices adequada terá um grande impacto na fase de recuperação. Por vezes, o que acontece é a base fornecer adequadamente proposições de casos quando o caso do usuário é comum, ou seja, não apresenta nenhuma novidade em termos dos atributos e de sua interligação. Contudo, novidades, isto é, casos anômalos, são exatamente o que uma boa base de casos deve indicar em sua composição. Isto é um problema, exige manutenção periódica da base e boa escolha de índices, como foi mencionado. Uma base excessivamente rígida não se comporta bem frente a casos que se aproximam do conhecimento armazenado mas apresentam pequenas porém importantes nuances. O necessário para contornar este problema é um maior poder de aproximação, ou de generalização dos valores dos atributos de caso existentes na base.

O segundo problema enfrentado também se relaciona a aquisição de conhecimento. A fase de adaptação repete uma situação comum no desenvolvimento de sistema especialistas

baseados em regras: nem sempre é fácil extrair regras de especialistas humanos. Além disso, é difícil manter o sistema de regras atualizado. As regras de reparo também apresentam esse problema, ainda que em menor escala, já que uma das vantagens de RBC é minimizar a aquisição de conhecimento. Essa redução porém acontece no que diz respeito ao modelo geral do problema. A necessidade de intervenção de especialistas ainda perdura quando é preciso moldar o caso do usuário segundo restrições ou heurísticas.

Conjuntos de regras de reparo são mais simples do que uma base de regras. Por exemplo, eles não são geralmente processados por motores de inferência, apesar de poderem ser encadeadas, mas mesmo assim, dependendo do domínio, apresentam alguma dificuldade em sua construção. Eles também, segundo algumas abordagens [Leake et al. 1995], podem ser na verdade um outro sistema de casos, associado à base, formando o que poderia ser chamado de uma *metabase* de casos. Esta *metabase* guarda não os casos relativos ao domínio desejado, mas uma coleção de operações de adaptação bem sucedidas relacionadas à base. Quando a fase de adaptação começa, o caso novo não é modificado segundo uma seqüência "cega" e automática de regras, e sim repetindo uma série de operações similares às que foram bem sucedidas anteriormente. Esta abordagem não foi incluída no software desenvolvido ao longo desta tese, mas são consideradas objetos de trabalhos futuros.

De qualquer forma, mesmo abordagens baseadas não somente em regras envolvem um grau de dificuldade na aquisição de conhecimento. Isto é inerente à idéia da adaptação em si. O desejado para reduzir estas dificuldades decorrentes da adaptação é um maior poder de extração de características a partir de um conjunto de dados disponível, ou também, maior poder de generalização dos atributos de solução disponíveis na base.

Ambos os problemas, portanto, parecem ser adequadamente tratados por sistemas que generalizam. A proposta é empregar um sistema conexionista, uma RNA, implementando uma memória associativa que execute duas funções durante a fase de recuperação e a de adaptação:

Associe cada um dos vetores de **atributos de caso** dos casos da base a um vetor de **atributos de solução**.

Transforme os valores destes vetores de atributos de solução de forma a exprimirem uma relação de diferenças entre o caso apresentado e o caso da base sendo selecionado.

A introdução desta RNA atenua a aquisição de conhecimento de um modo geral, em detrimento, sem dúvida, de uma maior clareza ou explicitação dos processos envolvidos. A descrição dos procedimentos pelos quais essa implementação se dá e da memória associativa propriamente dita encontra-se nas próximas seções deste capítulo.

### 3.3.1 Memória Associativa

A idéia básica que está por trás da introdução de um modelo conexionista na etapa de recuperação do ciclo RBC é a de implementar um método de reconhecimento de padrões capaz de identificar corretamente similaridades entre dois vetores de atributos.

Para que possa ser obtida também uma segunda capacidade, executar uma adaptação inicial, este método precisa também ser capaz de completar o padrão reconhecido, de modo que um vetor completo seja sempre obtido. A solução proposta para este problema é utilizar uma Memória Associativa.

A denominação Memória Associativa foi empregada para descrever o modelo responsável pela hibridização introduzida no par recuperação/ revisão. Este termo possui uma estreita relação com o conceito de Aprendizado Associativo, podendo mesmo vir a suscitar dúvidas quanto à adequação de seu uso.

A idéia de Aprendizado Associativo carece de uma definição precisa, sendo geralmente adotado para métodos de aprendizado conexionistas em que as sinapses (conexões) tendem a fortalecer-se à medida que aumenta a atividade entre seus dois nós formadores. Às vezes, esse aprendizado também é chamado de Hebbiano, em virtude da semelhança com o modelo apresentado por Donald Hebb [Hebb, 1949]. Porém, o termo associativo pode ser preferível, uma vez que a aplicação deste comportamento acaba por divergir da idéia original [Grossberg and Levine, 1987].

O modelo adotado neste trabalho inspirou-se no trabalho de Igor Tetko [Tetko, 2001], que optou pela mesma denominação, Memória Associativa, em seu artigo. Alguns autores estabeleceram o uso do termo a partir de trabalhos que se tornaram bastante disseminados. Entre eles merecem destaque o trabalho de Bart Kosko, apresentando o modelo BAM (Binary Associative Memory) [Kosko, 1988] e alguns dos principais artigos de Teuvo Kohonen [Kohonen, 1977]. Kohonen usou o termo Memória Associativa e Aprendizado Associativo distintamente para dois grupos de fenômenos afins.

O primeiro grupo, Aprendizado Associativo, envolve memória ou aprendizado que são desenvolvidos, por exemplo, por condicionamento clássico [Pavlov, 1927] ou aprendizado serial de listas. O segundo grupo, Memória Associativa, pode ser empregado para descrever processos em que um padrão armazenado é relembrado a partir de uma fração sua. Kohonen ainda prosseguiu, diferenciando aspectos das Memórias Associativas, referindo-se a memórias autoassociativas e heteroassociativas.

O primeiro tipo foi definido como uma operação em que um padrão incompleto é reconstruído em sua versão original, armazenada. Uma heteroassociação, em uma definição

complementar, seletivamente produz um padrão de saída $\mathbf{y}_k$ em resposta a um padrão de entrada $\mathbf{x}_k$. Neste caso, os vetores que compõem o par que associa $\mathbf{x}_k$ e $\mathbf{y}_k$ podem ser escolhidos livre e independentemente um do outro [Kohonen, 1984].

A escolha do termo nesta Tese segue esta mesma concepção: uma memória heteroassociativa.

Este modelo de Memória Associativa é treinado de forma a fazer a correspondência entre cada caso na base e um vetor de comprimento fixo. O primeiro algoritmo escolhido, para começar com a implementação de um modelo conhecido, foi uma rede MLP treinada com o algoritmo *backpropagation* [Haykin 1999]. Além dessa abordagem inicial, também foi estudada a hipótese da utilização de um outro modelo associativo, uma rede ART2 [Carpenter, Grossberg 1987]. Essa escolha foi influenciada pela possibilidade de comparação do primeiro com as características de aprendizado não supervisionado e baseado em instâncias do segundo. O modelo ART2 talvez seja mais natural para o processo desejado, uma vez que implementa um algoritmo que sempre atribui uma categoria a qualquer estímulo, constituindo-se, portanto, em um classificador eficiente. Essas vantagens já foram exploradas em hibridizações envolvendo apenas a etapa de indexação [Fabiunke et al. 1997]. Contudo, o foco deste trabalho desloca-se para a fase de revisão, durante as operações de adaptação. Nesta fase é comum surgirem ambigüidades nos padrões a serem manipulados (descrições dos casos). Casos, muitas vezes, envolvem fatores subjetivos, que regras locais podem tratar, ou que se comportam como se fosse ruído quando expostos a redes que empreguem aprendizado por *backpropagation*. Uma memória associativa que utilize ART2 apresenta dificuldades no trato de ambigüidades [Levine 1991] e necessitaria de uma análise de possíveis mecanismos auxiliares. A introdução deste segundo modelo de RNA no sistema e a conseqüente análise de comportamento escapa ao cronograma desta tese, constituindo-se porém em um interessante trabalho futuro.

### 3.3.2 Representação Adotada

A camada de entrada da rede MLP possui tantas unidades quanto for necessário para codificar os atributos dos casos armazenados na base. De modo a poder representar os casos nos scripts do ambiente de desenvolvimento RBC, esses atributos podem ser expressos em um dos cinco tipos:

- numérico (real ou inteiro);
- enumerado (discreto);
- listas;

- strings;
- booleanos.

Atributos numéricos e *booleanos* necessitam de uma unidade para cada valor representado. Atributos enumerados precisam ser previamente codificados por um algoritmo que leve em consideração a existência ou não de similaridades internas, como acontece, por exemplo, quando há ordem na enumeração. Esta informação de similaridade é incluída na base de casos. Geralmente, atributos enumerados precisam de tantas unidades quanto for a cardinalidade da enumeração. Outra alternativa seria o emprego de codificações padronizadas, como, por exemplo, códigos do tipo tons de cinza, que preservam a ordem dos valores. O Rabeca, porém, não representa os valores enumerados desta forma.

Atributos expressos por listas podem conter qualquer um dos outros tipos, e eles serão tratados do mesmo modo com se não fossem membros de uma lista. Atributos do tipo *string* são empregados apenas como informação auxiliar, não sendo aproveitados no treinamento da rede.

Após a conversão e a normalização dos atributos, eles são apresentados à camada de entrada da RNA.

A camada de saída da RNA possui um número fixo de unidades, dividido em duas áreas. A primeira área é composta por oito unidades e sempre representa o mesmo grupo de atributos da solução de um caso: o modelo escolhido (MLP, RBF, SOM) [Haykin 1999], a percentagem do conjunto de dados reservada para teste, a percentagem reservada para validação, o número de épocas entre as validações, a função de ativação, a função de iniciação, a função de atualização e a taxa de aprendizado (ou a taxa correspondente no caso de SOM's).

Em seguida a esta área, vem um grupo de cinco unidades que contém valores que podem variar de significado. É importante observar que, do ponto de vista da memória associativa, esses significados não fazem diferença. Após a associação, são usadas regras que transformam os números nos parâmetros respectivos, de acordo com o modelo escolhido. Dessa forma, a primeira unidade deste grupo indica a taxa de momento em um modelo MLP, mas pode conter o valor do raio de vizinhança em um modelo SOM. A divisão da camada de entrada é ilustrada na Figura 3.3.

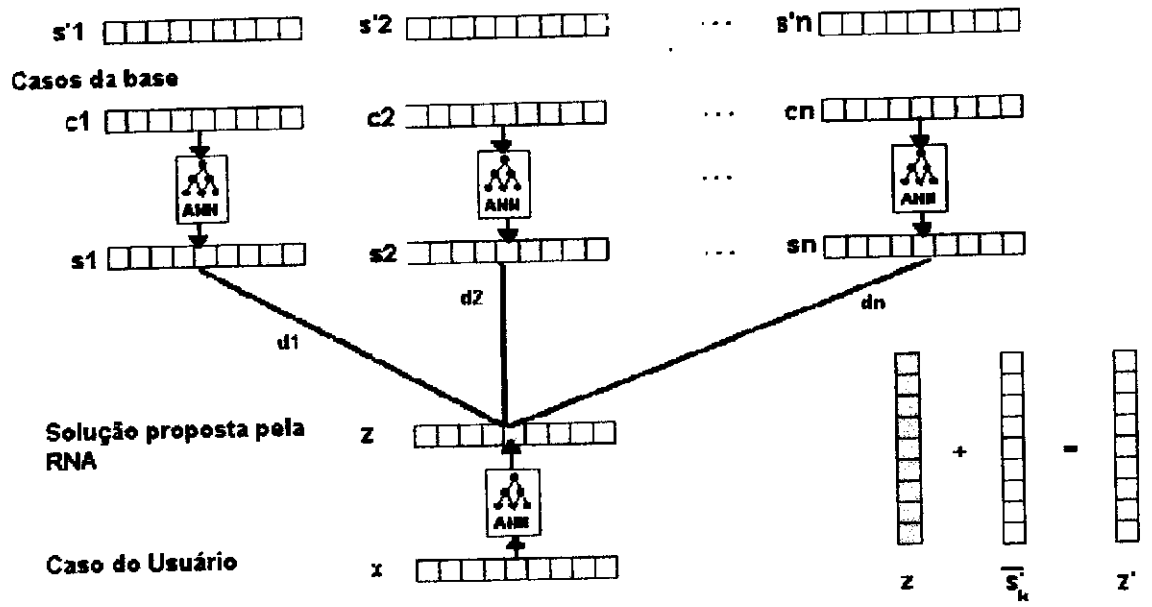


Figura 3.3 – Funcionamento Básico da Memória Associativa.

A camada oculta foi mantida em um tamanho (número de unidades) variando entre a quantidade de atributos de caso dos casos armazenados e a quantidade de unidades da camada de saída. Esta determinação ainda foi realizada de forma empírica.

### 3.3.3 Problemas e Soluções

Apesar das vantagens esperadas, o emprego de uma RNA como método de indexação do ciclo RBC apresenta dois novos problemas. O primeiro deles é a necessidade de retreinamento periódico da RNA. Isto acontece porque a rede aprende a reconhecer a base de casos inicial, apresentada quando a rede é treinada pela primeira vez. À medida que o sistema completo (o sistema RBC ) começa a aprender novos casos, este mecanismo tende a tornar-se impreciso. Deve ser notado, porém, que este problema, a manutenção da base de casos e do sistema de indexação, também existe em abordagens não conexionistas, uma vez que o crescimento da base necessita ser controlado de forma a garantir um bom balanceamento (nenhum conjunto de atributos cosntituir uma porcentagem exagerada da base) e representatividade (casos realmente comuns e importantes, protótipicos, possuirem ao menos um exemplar).

O segundo problema advém da impossibilidade de um modelo baseado somente em uma RNA em estabelecer um *ranking* de soluções. A RNA pode escolher um caso da base, mas não pode proporcionar uma segunda ou terceira sugestões.

De modo a encontrar uma solução para ambos os problemas, foi adotado um modelo de memória associativa inspirada em Tetko [Tetko, 2001]. Neste modelo, dois métodos trabalham em paralelo:

- um *ensemble* de RNAs, ou seja, um conjunto de RNAs sendo ativadas ao mesmo tempo cujas respostas serão analisadas ao término de sua execução para a determinação de uma resposta global. Esta resposta global pode ser a média das individuais, por exemplo. A idéia na utilização de *ensembles* é minimizar a influência do estado inicial dos pesos de cada rede.
- um algoritmo de vizinho mais próximo ( *nearest neighbor,* k-NN).

O resultado apresentado pelo algoritmo k-NN é usado para dois fins :

- ajustar a saída da RNA;
- proporcionar um *ranking* opcional, no caso do primeiro caso selecionado não ser aprovado pelo usuário.

O funcionamento básico da memória associativa é mostrado na Figura 3.3.

### 3.3.4 Funcionamento Básico

Segundo as sugestões da memória associativa proposta, a saída da RNA (**z**) proveniente do caso apresentado pelo usuário (**x**) não é o valor definitivo apresentado pela associação. Um grupo de **n** casos (selecionados inicialmente de acordo com os atributos índice da base) é fornecido como entrada à primeira camada da RNA, gerando um conjunto **s** de vetores. Estes vetores deveriam ser iguais aos das soluções armazenadas na base de casos, caso a RNA operasse sem introduzir erros. Quaisquer discrepâncias que ocorram são guardadas em um conjunto de vetores **s'**. Variáveis contínuas simplesmente têm suas diferenças calculadas. Atributos discretos são submetidos a um breve ajuste para que sejam levadas em conta questões referentes à proximidade de valores dentro de enumerações ou generalizações.

Neste ponto, o algoritmo k-NN já é capaz de montar seu *ranking* de casos, calculando a distância **d** entre cada vetor **s** e o vetor **z**.

Contudo, a solução final proposta será o vetor **z'**, definido pela soma do vetor **z** com o vetor $\overline{\mathbf{s}'}_{\mathbf{k}}$, contendo a média aritmética dos **k** vetores mais próximos **s** encontrados no *ranking* do algoritmo k-NN.

A correção proporcionada pelo algoritmo k-NN cria um segundo nível de conhecimento, talvez melhor descrito como conhecimento local, complementando o conhecimento global armazenado na RNA.

A RNA realiza uma primeira seleção. Esta seleção será tão melhor quanto maior for a proximidade da base de casos corrente com a base de casos originalmente utilizada para treinar-se a RNA. À medida que a base de casos cresce e a RNA permanece sem um novo treinamento, a correção fornecida pelo algoritmo k-NN torna-se maior. Pode ser estabelecido um valor máximo para esta correção de modo que ao ser atingido, um novo treinamento seja realizado, porém este processo não terá que ser executado com freqüência. Mesmo quando a RNA tiver sido treinada recentemente, espera-se que o ajuste do algoritmo k-NN seja tão mais influente quanto maior tornar-se a base de casos, em virtude de uma complexidade maior do espaço de soluções.

Além disso, o ranking de casos construído pelo algoritmo k-NN pode ser usado para reescolher casos, se necessário. Cada caso do ranking já foi apresentado à camada de entrada da RNA, gerando um vetor de saída (s) e pode ser usado para prosseguir no ciclo de RBC.

O algoritmo k-NN implementado no RaBeCa utiliza distâncias euclidianas como medida de similaridade.

A representação dos vetores na camada de entrada da RNA também necessita tratar o problema de ausência de valores na base de casos. Estes valores ausentes ocorrem devido a não definição de atributos por parte do usuário. O algoritmo de codificação da camada de entrada trata essas ocorrências de duas formas diferentes. Atributos que não estejam presentes no caso ou que recebam o valor de 'não aplicável' pelo usuário, são tratados de uma mesma forma: eles são considerados como não possuindo nenhuma influência na seleção. Atributos marcados como 'desconhecido' são tratados de outra maneira. Eles são considerados existentes e influentes.

Os valores 'desconhecidos' são substituídos pela média aritmética (ou pela moda, em campos discretos) de todos os valores daquele atributo que apareçam na base de casos. Tanto o algoritmo k-NN quanto a camada de entrada da RNA operam desta forma.

Os atributos 'não aplicáveis' forçam o algoritmo k-NN a estabelecer uma distância igual a zero para cada um deles. Contudo, a camada de entrada da RNA precisa receber um valor significativo. Este valor é a média aritmética (ou moda) calculada sobre os valores dos atributos da base, porém levando-se em conta apenas aqueles valores que apareçam ao mesmo tempo com os demais valores fornecidos pelo usuário. Para que uma ocorrência de um determinado valor entre no cálculo da média, ele tem que aparecer em casos que possuam os demais atributos instanciados com valores iguais aos do caso do usuário.

## 3.4 Considerações Finais

A proposta de abordagem híbrida, apresentada neste capítulo, procura prosseguir na idéia de que para um sistema inteligente é vantajoso, para a utilização prática no mundo real, que ele seja capaz de absorver conhecimento do mundo, ou, em outras palavras, bom senso. Por isso, procurou-se caracterizar um arcabouço possível para a hibridização, descrever e justificar a construção de uma Memória Associativa e assinalar os principais aspectos desta integração neuro-simbólica, que pretende, justamente, dotar o modelo de uma capacidade maior de generalização. A integração deste modelo em um sistema que use RBC poderá ser mais bem compreendida após a apresentação do ambiente desenvolvido para tal, o que será o assunto do capítulo seguinte.

# Capítulo 4

# RABECA: O Ambiente de Desenvolvimento

A motivação principal para a criação do ambiente RaBeCa foi a possibilidade de se dispor de um *software* aberto o suficiente para que fossem testadas diversas possibilidades de inclusão e mesmo hibridização com outros paradigmas comuns em sistemas inteligentes.

A determinação dos pontos em que essa hibridização poderia ser realizada é um dos objetivos da pesquisa. Inicialmente, e a título de esclarecimento das contribuições do trabalho, existem três idéias para sua aplicação. Em primeiro lugar, durante a fase de recuperação. Seria introduzida uma RNA no mecanismo indexador, o qual foi mais bem explicado no capítulo anterior. Essa rede implementa uma memória associativa que liga o conjunto dos valores dos atributos descritores do caso proposto ao caso mais semelhante existente na base, ou mais adequadamente, a um chamado caso-protótipo, espécie de identificador de uma classe à qual pertenceria o caso proposto.

Essa é exatamente a função do indexador, realizada geralmente através de algoritmos não adaptativos. Porém, a adoção de uma RNA para a realização desta tarefa, além de conferir maior maleabilidade ao processo em geral, possibilita o uso de uma gama mais extensa de tipos de atributos na descrição do caso (imagens, dados em tempo real, massas de dados numéricas mais volumosas). Um maior poder de representação de atributos pode ser conseguido pela introdução direta deste tipo de dados (imagens, sinais, grupos de atributos) na camada de entrada de uma RNA. Na prática o que acontece é um pré-processamento do caso, contornando, por exemplo, problemas de atributos desconhecidos em domínios onde os valores estão disponíveis não ao conhecimento do usuário, mas a leituras diretas por sensores. No domínio abordado nesta tese, este tipo de atributo não foi considerado. Por conseguinte foi dada prioridade ao aumento da capacidade de generalização conseguido pelo mecanismo de indexação.

Os algoritmos de indexação, em geral, necessitam de um criterioso estabelecimento de pesos em seus parâmetros de modo a levar em consideração os atributos da forma que o

usuário realmente deseja. Esse processo pode ser replicado dentro da RNA, exigindo uma participação menor do usuário, uma vez que a rede poderia ser treinada regular e automaticamente pelo próprio sistema conforme os casos da base sejam alterados.

Uma segunda aplicação seria estender essa ação da RNA à fase de revisão. O processo de classificação executado pela rede não teria necessariamente que se ater à escolha do caso existente, podendo já associar o caso proposto a casos mais completos. Assim, se o caso proposto não tiver todos seus atributos fornecidos, ou esses atributos não forem exatamente os esperados pela base, o processo de adaptação talvez possa dar-se ao mesmo tempo que o de indexação, com as informações não existentes sendo estimadas pela própria habilidade de associação neural, ou seja, o padrão associado na saída já seria o completo. Essa hibridização seria o análogo do que acontece quando se constrói um sistema especialista conexionista, com uma base de regras de produção sendo substituída por uma RNA. O processo de adaptação (e o pré-processamento também) é realizado através da aplicação de regras pré-estabelecidas e que, consequentemente, exigem uma atenção maior em sua manutenção e atualização.

Finalmente, um terceiro ponto merecedor de estudo seria a fase de retenção de um caso. Em bases de casos pequenas, o problema de sua manutenção é menor, ou mesmo inexiste. Contudo, à medida que se acumula experiência na base, e essas são as bases que serão empregadas na maioria dos sistemas RBC realmente bem sucedidos na prática, a decisão de manter ou não um caso na base, ou incluir um caso novo, torna-se difícil para o usuário. O problema aqui é a garantia de uma base de casos extensa que seja homogênea e representativa, ou seja, com uma distribuição de casos não tendenciosa em que os casos que realmente sintetizam situações básicas importantes estejam presentes. A partir de uma certa dimensão, esse é um problema de classificação [Smyth, McKenna 1998]. Ainda que ferramentas para manutenção possam basear-se em medidas estatísticas, mais uma vez um classificador conexionista poderia apresentar vantagens quanto ao manuseio e adaptabilidade. RNAs não supervisionadas podem ser empregadas nesse processo, juntamente com um critério simples de inclusão baseado nas classes encontradas.

O segundo objetivo desta tese é a aplicação do método de RBC no próprio projeto e implementação de RNA, de modo a construir-se um programa que oriente o usuário, leigo ou não, quando do desenvolvimento de um novo sistema conexionista. Assim, o ambiente RaBeCa é apenas a primeira etapa nessa direção. Contudo, em sua implementação, a idéia básica é a de que ele seja tornado público, ao menos no que diz respeito ao código executável, e permaneça passível de inclusões variadas, não só pela própria equipe envolvida em seu projeto, mas através de contribuições externas. Para esse fim, sua arquitetura dispõe de pontos

específicos onde novas rotinas podem ser incluídas, segundo um padrão de interface. Este padrão ainda está em fase de definição e a tendência é o uso de bibliotecas dinâmicas para comunicação entre módulos executáveis e expansões na linguagem descritora das bases de casos que permitam a inclusão de novas primitivas.

## 4.1 Principais Componentes do Ambiente

O ambiente Rabeca é constituído de quatro partes principais: um interpretador, um pré-processador, um indexador e uma interface com o usuário, considerando-se que a base de casos é montada dinamicamente. A Figura 4.1 mostra o diagrama de interligação dos principais blocos constituintes do sistema.

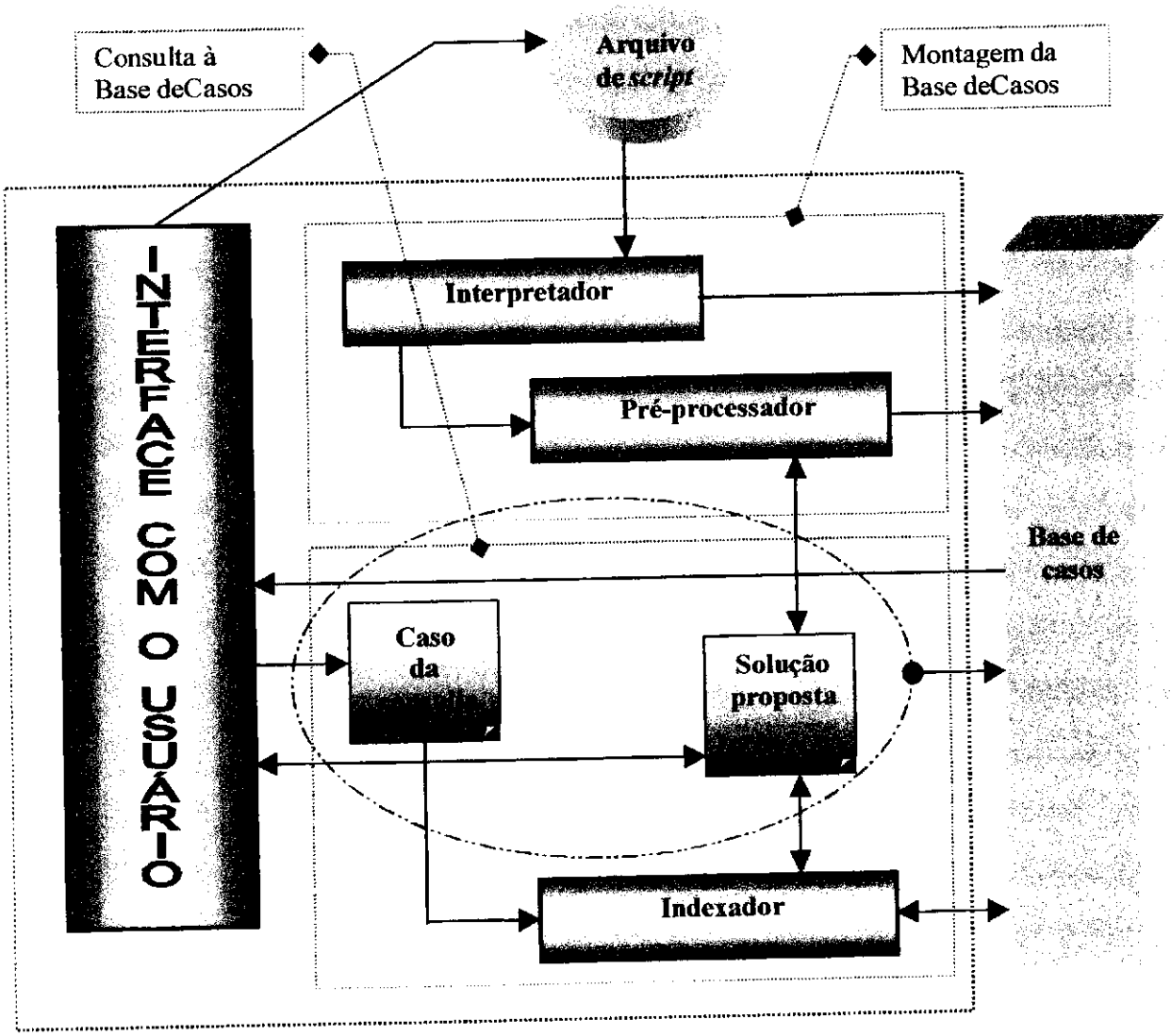


Figura 4.1 - Diagrama de Interligação dos componentes do RaBeCa

O interpretador do RaBeCa aceita descrições de casos seguindo uma linguagem de *script* compatível com CASL, linguagem criada pela Universidade de Wales para gerar a base

de casos para seu próprio ambiente de RBC, CASPIAN [Pegler, Price 1996]. Apesar de basear-se nessa linguagem, o interpretador do RaBeCa foi construído independentemente e inclui extensões não disponíveis na versão original da linguagem. Atualmente, essas extensões referem-se à possibilidade de inclusão de uma RNA como representação de conhecimento na base de casos e comandos para a execução de programas externos a partir da solução encontrada. Esta alternativa tem por objetivo possibilitar o encadeamento do sistema de RBC com outros processos externos que implementem partes da solução proposta. A extensão anterior, por sua vez, permite ao usuário determinar, automaticamente, valores para atributos que ele pode não conhecer com exatidão. Por exemplo, o campo do atributo pode ser a classificação de um conjunto de entradas provenientes de sensores ou o resultado de uma função desconhecida ou complexa, que seja melhor expressa por uma RNA. Porém, pretende-se que a extensão da linguagem seja um dos pontos de desenvolvimento futuros.

```
introduction is

'\t\tCHEF DEMO 2\n\t\t--- --- -\n',
'Esta é uma versão modificada do demo CHEF, que mostra',
'as características da linguagem CASL versão 49 usada pelo' RaBeCa. É
uma ' 'tradução aproximada do original em inglês',
'referenciado no livro "Case-Based Reasoning" de autoria da Dra. Janet '
'Kolodner e usado como demonstração do CASPIAN também'
end;

~Aqui são definidos os campos usados na seção de descrição do caso

case definition is

field forma_de_preparo type is (cozer, assar, fritar)
prompt is ['Entre a forma de preparo:'];
field sabor type is (apimentado, adocicado, temperado, acridoce)
prompt is ['Escolha o sabor do prato'];
field ingredientes type is list
prompt is ['Forneça alguns dos ingredientes que deseja usar.'];
end;

~Deve haver no mínimo um campo definido como indexador.
~Campos indexadores têm que ser enumerações

index definition is
index on forma_de_preparo;
end;
```

Figura 4.2 - Trecho inicial de script em CASL

Basicamente, segundo sua sintaxe, o *script* em CASL possui seis seções principais:

- identificação da base;
- definição dos atributos dos casos;
- índices;
- modificações;
- regras;

- instâncias

A primeira seção apresenta uma identificação da base, contendo apenas um texto formatado de conteúdo livre. Em seguida, vêm a definição dos atributos básicos do caso a ser proposto e a definição dos índices. Nessas seções são definidos os nomes dos campos, seus tipos e os atributos que serão usados como indexadores iniciais, isto é, os atributos considerados como fundamentais na primeira tentativa de escolha de um caso por similaridade. A Figura 4.2 ilustra esses dois primeiros blocos. A Figura 4.6 mostra seu resultado dessas escolhas na interface com o usuário.

```
~esta seção define uma hierarquia de símbolos, que especifica que
~símbolos são similares. Além disso, símbolos abstratos casam com
~seus derivados em regras de reparo e quando da escolha de casos

~Esta seção também é usada para definir-se intervalos para campos
~numéricos, usados para definir similaridade entre números

modification definition is
   abstraction carne_vermelha is (vaca, porco);
   abstraction carne_branca is (frango);
   abstraction qualquer_carne is (carne_branca, carne_vermelha);
   abstraction qualquer_vegetal is (ervilhas, brócolis, vagem);
   abstraction verdura is (vagem, brócolis);
   abstraction legume is (ervilha, vagem);

   abstraction tipo_de_queijo is (queijo, tofu);
   abstraction molho is (soja, madeira);
   abstraction vinho is (vinho_de_arroz, vinho_tinto, vinho_branco);
   abstraction tipo_de_amido is (maisena, amido);
   abstraction adoçante is (açúcar, aspartame);
   abstraction pimenta is (pimenta_vermelha, pimenta_verde, chili);
end
```

Figura 4.3 - Trecho de *script*, Seção de Modificações.

A seção de modificações é usada para especificar relações entre os diversos símbolos, visando à determinação de similaridades. Abstrações são símbolos criados como novos atributos e que compreendem um subconjunto de atributos declarados na definição do caso, ou seja, um símbolo que pode ser substituído por qualquer de seus associados. Esta seção também é usada para definir intervalos de similaridade numérica, para atributos inteiros ou reais. Um exemplo dessa seção é fornecido na Figura 4.3.

Duas seções contêm as regras de adaptação que podem ser aplicadas pela base: a seção de regras de pré-processamento e a de regras de reparo. Essas regras serão mais bem explicadas mais adiante no texto, quando for descrito o comportamento do pré-processador, mas um breve exemplo é apresentado na Figura 4.4.

```
~regras de pré-processamento são aplicadas logo após o usuário
~ter fornecido os parâmetros de busca. Elas funcionam como as
~regras de reparo, mas não permitem que o caso original seja
~re-escolhido

preprocess rule definition is

~Esta parte demonstra o processo de extração, que funciona
~procurando numa lista (no exemplo, ingredientes) por um
~símbolo que é uma especialização da símbolo abstrato fornecido
~(tipo_de_queijo, molho, etc.).

~O símbolo na lista poderia ele próprio ser um abstração de outro
~símbolo se forem usados múltiplos níveis de abstração. Por
~exemplo, se a ação fosse "extract qualquer_carne" e o usuário
~pusesse carne_vermelha na lista de ingredientes, esse valor
~seria removido.

~Se não existe nada na lista que seja uma especialização
~do símbolo fornecido, o valor anterior é mantido.

~esta regra cria campos individuais novos para abrigar cada um
~dos ingredientes especificados pelo usuário

repair rule separa_ingredientes is
when
1 = 1 ~execute esta regra sempre !
 then
   change carne extract qualquer_carne from ingredientes;
   change ingr1 extract tipo_de_queijo from ingredientes;
   change ingr2 extract molho from ingredientes;
   change ingr3 extract vinho from ingredientes;
   change ingr4 extract tipo_de_amido from ingredientes;
   change ingr5 extract adoçante from ingredientes;
   change vegetal extract qualquer_vegetal from ingredientes;
   change ingr7 extract tempero from ingredientes;
   end;
end;
```

Figura 4.4 - Regras de Pré-processamento

A última seção contém os casos da base propriamente ditos. Cada caso, chamado de instância, divide-se em duas partes: o caso em si, descrito por pares de atributos-valores, e a solução associada. Os atributos presentes nos casos sempre possuem referências nas seções anteriores. As soluções podem apresentar novos termos ou mesmo textos explicativos, sendo uma das alterações introduzidas no ambiente RaBeCa a possibilidade de um campo da solução ser um comando de execução de um programa externo. Um breve exemplo de uma descrição de caso encontra-se na Figura 4.5.

Essas seções compreendem todos os itens necessários para a montagem da base de casos. Na versão aceita pelo RaBeCa, além das extensões mencionadas, incluiu-se também uma versão em português do conjunto de palavras reservadas, de modo a tornar a sintaxe mais clara para usuários não habituados com o uso de inglês.

O pré-processador, nesta etapa, carrega as regras de transformação (aqui, regras de pré-processamento) que irão ser eventualmente empregadas quando da recuperação de um caso. Ele também é acionado para as demais regras, chamadas regras de reparo, como será visto adiante. Uma das idéias testadas, conforme mencionado anteriormente, é a introdução de uma Rede Neural também ser introduzida neste ponto, de modo a permitir transformações fundamentadas em um conhecimento não inteiramente explícito. Contudo, a primeira abordagem considerada, o uso de regras de produção, permanece opcional ao usuário.

As regras de pré-processamento são análogas tanto em ação quanto em sintaxe àquelas que mais tarde serão empregadas na fase de revisão do caso, as regras de reparo. A diferença é que essas, como seu nome indica, modificam o caso proposto. O caso proposto é aquele apresentado pelo usuário, antes que ele seja passado ao indexador para uma consulta à base de casos existente e cause o retorno de uma primeira instância de solução possível. Essas regras tentam moldar o caso proposto às restrições gerais já estabelecidas pelo contexto da base. Elas modificam atributos, podendo extrair valores de listas, alterar pesos, executar operações aritméticas, exibir mensagens na tela, selecionar novos casos e alterar valores de atributos.

As regras de reparo, por sua vez, são acionadas após um caso ter sido recuperado e executam o mesmo tipo de ação que as regras de pré-processamento. Elas podem ser locais, isto é, específicas de um determinado caso, ou globais, válidas para todos os casos selecionados. Se forem locais, só serão acionadas se o caso a que pertencem for escolhido eaparecem dentro da definição da instância (descrição do caso na base). Uma vez que a base de casos esteja montada na memória, o sistema está em condições de ser consultado. Para tal, a interface com usuário apresenta um diálogo, isto é, uma área para troca de informação, construída a partir dos atributos incluídos na base, onde o usuário pode descrever seu caso para consulta. Atualmente, os atributos disponíveis podem ser valores numéricos, reais ou inteiros, valores enumerados, valores lógicos (booleanos) e *strings* livres. Estes últimos não podem ser usados como índices para escolha de casos. A Figura 4.6 mostra a tela básica do sistema após o carregamento de uma base, juntamente com uma amostra de diálogo para configuração de caso e consulta à base. O caso assim apresentado é passado ao módulo indexador que, utilizando-se de algoritmos de classificação, fornece uma lista dos casos existentes na base mais próximos ao apresentado. O RaBeCa, no momento, dispõe de uma variação ponderada do algoritmo de vizinho mais próximo (*nearest neighbour, k-NN*) [Dasarathy 1990] e da memória associativa implementada através de uma RNA ( MLP ), conforme visto no capítulo anterior.

```
~a seguir vêm os casos
~apenas o primeiro caso foi incluído

case instance frango_frito_com_vagem is
        forma-de_preparo = fritar;
        ingredientes = [frango, vagem];
        sabor = acridoce;
        carne = frango;
        vegetal = vagem;
solution is
        preparo_da_carne = desossar;
        preparo_do_vegetal = descascar;
        receita = [[preparação_da_carne carne]
        [preparação_do_vegetal vegetal]
        [forma_de_preparo carne 'e o' vegetal 'juntos'
        'usando uma panela wok']];

local repair rule definition is
        repair rule reescolhe_1 is
        when
                carne is undefined
        then
                pr(['Abandonando  caso porque a carne não está definida']);
                reselect;
        end;
        repair rule reescolhe_2 is
        when
                vegetal is undefined
        then
                pr(['Abandonando caso porque o vegetal não está definido']);
                reselect;
        end;
        repair rule preparo_do_brócolis is
        when
                vegetable is brócolis
        then
                change preparo_do_vegetal to picar;
                change hd(tl(tl(tl(tl(tl(hd(tl(tl(receita)))))))))
                to separadamente;
                pr(['Aviso: A carne de' carne 'deixa o' vegetal 'ensopado']);
                pr(['quando eles são fritos juntos, então cozinhe']);
                pr(['a carne de' carne 'e o' vegetal 'separadamente']);
        end;
        repair rule preparo_do_legume is
        when
                vegetal is legume
        then
                change preparo_do_vegetal to descascar;
        end;
end;
```

Figura 4.5 – Instância de Caso e Regras de Reparo Locais

### 4.1.1 Indexação empregada

Resumidamente, o funcionamento do indexador é descrito como um algoritmo capaz de consultar um conjunto de padrões (casos), confrontá-los com um novo padrão apresentado (caso do usuário) e decidir qual daqueles padrões armazenados lhe é o mais próximo, segundo um critério fornecido. Nos métodos mais simples, esse critério é uma medida de distância entre os diversos atributos, geralmente ponderados. Na versão inicial do RaBeCa foi empregado

apenas o algoritmo k-NN. Empregando-se este algoritmo, o indexador busca, em uma primeira tentativa, encontrar casos que possuam os atributos declarados como índices o mais próximos possível do caso apresentado. Essa proximidade é calculada por uma distância euclidiana ponderada pelos pesos dos atributos, com os atributos não numéricos recebendo uma representação adequada. Nas etapas de revisão, todos os atributos são levados em consideração.

Em sua versão final, o mecanismo indexador utiliza a memória associativa descrita no capítulo 3. O algoritmo k-NN a que o texto ali se refere é o mesmo da versão original. Portanto, a implementação do segundo processo de indexação foi realizada de forma incremental, ou seja, emprega os recursos já disponíveis e disponibiliza ambos os métodos como opções.

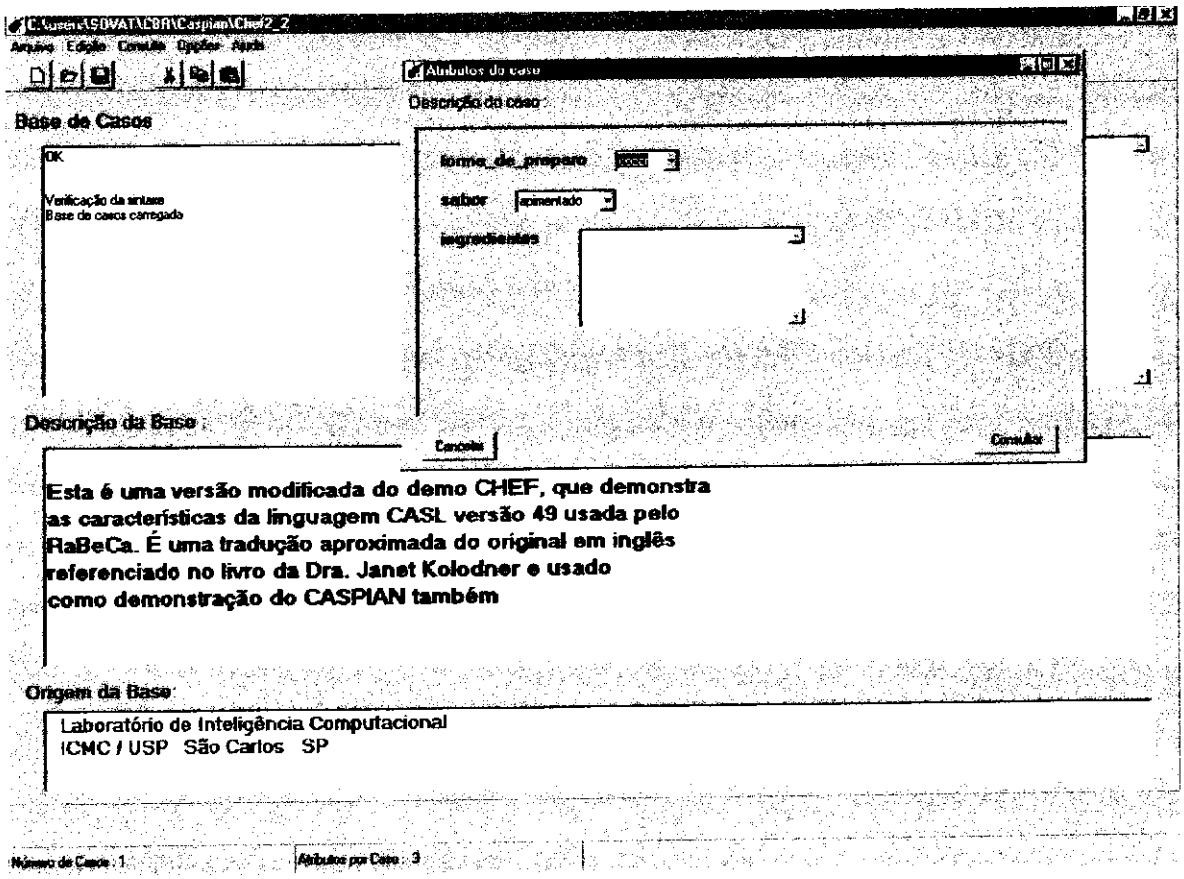


Figura 4.6 - Tela Básica do Sistema

No exemplo da Figura 4.6, existe apenas um índice, o atributo "forma_de_preparo". Supondo-se que o usuário apresentasse um caso com "forma_de_preparo" definida como "assar", o indexador retornaria o primeiro caso que também possuísse tal atributo com aquele valor. Em geral, os índices são múltiplos. O atributo índice pode ser também uma

generalização de um conjunto de valores. Neste caso, o indexador também retorna casos que apresentem valores pertencentes a especificidades do atributo enumerado.

O usuário pode, em seguida, iniciar o ciclo de revisão da solução proposta. Isto é realizado acionando adaptações do caso recuperado a partir de regras existentes no pré-processador, ou alterando os atributos diretamente na interface. O caso alterado é então reavaliado pelo usuário. Se ainda não for satisfatório, pode ser tentada uma nova indexação ou pode-se decidir pelo seu descarte, passando ao próximo caso da lista fornecida anteriormente. Essa reavaliação varia conforme os critérios do usuário e o domínio da aplicação desejada.

Na maioria das vezes, esse é um processo subjetivo ou complexo, realizado apenas através da interface por um especialista. Contudo, métodos automáticos estão sendo estudados para aplicações específicas [Netten, Vingerhoeds 1996].

Quando uma solução proposta é considerada satisfatória, é apresentada a possibilidade de inclusão do caso na base existente, de modo a aumentar o conhecimento do sistema ou simplesmente a gravação da solução para uso futuro. As soluções, aceitas ou não, podem conter referências a programas externos, de modo a permitir que ações sejam executadas a partir das mesmas. A interface permite ainda a criação de arquivos contendo os atributos descritores dos casos e instâncias a serem incluídas na base de casos. Pretende-se que futuramente toda a base possa ser criada a partir do próprio ambiente. No atual estágio de desenvolvimento, ainda é mais eficiente criar-se o *script* de montagem em um editor de texto externo.

### 4.1.2 Critérios de Similaridade

A idéia dominante durante o desenvolvimento desta tese foi a de que o ambiente RaBeCa possuísse critérios de similaridade variados. Contudo, devido a restrições de tempo para implementação, o que foi realizado consiste fundamentalmente em estabelecer critérios específicos e fixos para cada tipo de atributo. Um dos objetos componentes do RaBeCa é uma instância de uma classe que foi denominada TComparador, que executa um tipo de comparação para cada tipo de atributo, segundo esses critérios. Esses critérios são descritos a seguir, para cada tipo de atributo tratado no ambiente. Para todos os critérios, o índice de similaridade varia de 0 (nenhuma similaridade) até 1 (igualdade).

- Enumerações – Nessa primeira versão do RaBeCa, são sempre consideradas ordenadas. Primeiramente verifica-se se o atributo é uma abstração, ou seja, se ele foi incluído na seção de modificações e associado a um grupo de identificadores. Em caso positivo, o comparador irá considerar um índice máximo ($S_i$ = 1.0) se o valor do

atributo de um caso for igual ao do outro ou estiver na lista de generalizações (abstrações) do outro. Não possuindo uma abstração, o índice de similaridade será calculado pela fórmula 4.1:

$$S_i = 1 - \frac{\left| x_i^1 - x_i^2 \right|}{QtdInt} \qquad (4.1)$$

onde:

$S_i$ é o índice de similaridade relativo ao atributo de ordem $i$;

$x_i^n$ é o valor do atributo $i$ do caso $n$;

*QtdInt* é o total de intervalos para o atributo.No caso de enumerados, é dado pela cardinalidade da enumeração, isto é, pelo total de valores possíveis para a enumeração.

Numéricos – Da mesma forma que nas enumerações, primeiro verifica-se se o atributo está vinculado a algum intervalo de similaridade. Caso esteja, seu índice também é calculado pela equação 4.1. Se o atributo pertencer a mais de um intervalo, é levado em consideração o intervalo que possuir o menor limite inferior. Se o atributo não pertencer a nenhum intervalo, seu índice é calculado pela equação abaixo:

$$S_i = e^{-\left| \frac{x_i^1 - x_i^2}{x_i^2} \right|} \qquad (4.2)$$

para $x_i^2 \neq 0$

$$S_i = e^{-x_i^1}$$

para $x_i^2 = 0$

*Strings* – Têm seu índice de similaridade calculado pela proporção entre palavras iguais e o total de palavras na *string*. Contudo, este tipo de atributo ainda não é empregado para o cálculo da similaridade entre dois casos.

Listas – Nesta primeira versão apenas são consideradas iguais (S = 1.0) se cada atributo de uma lista, respeitando-se a ordem, for considerado igual ao da outra lista.

Booleanos – Só apresentam índice de similaridade 1 ou 0, a partir de uma operação de conjunção lógica (AND).

## 4.2 Considerações Finais

Um dos objetivos primários deste trabalho é a investigação das vantagens e desvantagens de mesclarem-se uma outra técnica de Inteligência Artificial, no caso RNAs, com o ciclo RBC. O RaBeCa não é encarado como um fim, mas como um meio de atingir-se uma maior capacitação na experimentação na máquina. Por exemplo, ainda que uma correta recuperação de um caso da base seja fundamental para todo o processo, é errôneo pensar que um ciclo RBC resume-se a um bom algoritmo do tipo *nearest neighbour*. Questões relativas a uma escolha adequada da representação dos casos, à introdução eficiente de um significativo senso comum e ao entendimento e representação de critérios empregados por usuários especialistas para a avaliação de soluções constituem uma oportunidade valiosa de experimentarem-se vertentes de diversos sistemas inteligentes. No Capítulo 8 alguns pontos de desenvolvimento futuros são indicados.

# Capítulo 5

# A Montagem da Base de Casos

A base de casos de RNAs foi codificada na linguagem de *script* que o Rabeca utiliza, uma extensão da CASL [Pegler, Price 1996]. Ela compreende não só a definição dos casos em si, mas também dos atributos índice, das generalizações que porventura sejam necessárias, das regras de pré-processamento e de reparo e, sobretudo, dos atributos do caso. A base é composta de 23 casos envolvendo os três principais modelos empregados em problemas de classificação (MLP, SOM e RBF). Nao apêndice A é mostrado um extrato dos casos e os elementos da base. Neste Capítulo, serão descritas as principais características levadas em consideração durante a construção da base.

## 5.1 Estrutura de um Caso

Os casos são formados por uma coleção de pares atributo/ valor, contendo informações julgadas relevantes para a definição de um dado problema e sua solução. A especificações fornecidas ao sistema constituem o **caso do usuário**. As soluções anteriores, armazenadas em memória, formam os **casos da base**.

Diferentemente do caso proposto pelo usuário, que explica apenas o que ele deseja fazer e descreve seus dados, os casos da base compõem-se de duas seções de atributos: os de caso e os de solução.

Os atributos de caso dos casos da base, assim como os do caso do usuário, necessariamente são um subconjunto dos atributos de caso estabelecidos pelo projetista para um determinado domínio. Esse subconjunto, dentro da sintaxe do RaBeCa, é chamado de **campos do caso**.

Atributos de solução são deixados em aberto para cada caso, não necessitando enquadrar-se nessas lacunas pré-determinadas pelo construtor da base (campos). Assim, cada solução nova integrada à memória pode trazer aspectos específicos de sua implementação. Isso permite a introdução de novos paradigmas e variações sobre os modelos tradicionais, além de

um grau maior ou menor de detalhamento de cada solução, conforme tenha sido possível ou não se obter uma descrição detalhada.

Uma base de casos referente ao projeto de RNAs apresenta a característica de possuir muito mais atributos de solução do que de caso. Por conseqüência, a etapa de adaptação do caso recuperado será mais importante nesta base do que é usual em RBC. A adaptação será realizada diretamente pelo usuário, editando os valores necessários na tela, e pelas regras de reparo.

Os atributos da solução serão compostos pelos parâmetros da rede, por informações relativas ao pré-processamento e por estratégias a serem seguidas durante seu treinamento.

Os parâmetros da rede podem ser os valores numéricos usados para definir a rede e controlar o treinamento (taxa de aprendizado, termo de momento, topologia da rede) ou as escolhas das funções ( ativação, iniciação, cálculo do erro). O pré-processamento indica o tipo de normalização de dados efetuada (linear, logarítmica) e as eventuais escolhas de representação de dados (quantas unidades de entrada por valor, conversões e ordenações de campos enumerados). As estratégias de treinamento envolvem a proporção de partição do conjunto (treinamento, validação e teste) e a escolha do algoritmo de treinamento.

## 5.2 Principais Parâmetros

Entre as diversas considerações surgidas durante o desenvolvimento do trabalho, uma das questões foi justamente determinar até onde construir o arcabouço sobre o qual deverão ser descritas as RNAs. Essa descrição deve ser livre o suficiente para que casos novos ou variações mais notáveis de casos antigos possam ser aceitas, enquanto que, ao mesmo tempo, evite a criação de redes inviáveis ou não factíveis. Assim, também é parte do trabalho uma etapa de Engenharia do Conhecimento, em que a visão de projetistas experientes seja introduzida na base de casos, até mesmo porque se espera que usuários leigos beneficiem-se do potencial de treinamento por trás do sistema final que está sendo descrito neste trabalho.

A questão sendo enfrentada, então, é a de como equilibrar adequadamente essas duas restrições: um modelo fraco e um sistema coerente. A abordagem inicial pretende que um grupo reduzido de atributos seja escolhido pelo usuário, deixando a adaptação dos casos para as regras. Como proposta apenas para o conjunto dos atributos necessários à definição do caso, esses atributos poderiam ser :

tipo de aplicação : a idéia é de que este atributo seja usado como um dos índices para escolha de casos. Inicialmente, os valores possíveis previstos são classificação, regressão, associação e otimização.

• arquivo da amostra : uma vez que os dados a serem processados pela RNA estejam disponíveis para o sistema, uma avaliação dos parâmetros estatísticos básicos pode ser feita. Esses parâmetros seriam mais importantes na indicação de similaridade de domínios do que propriamente na análise do desempenho possível e da qualidade da resposta da Rede Neural. A disponibilidade desse material determinaria também o tipo de avaliação da solução. Uma vez que a rede pode ser executada logo em seguida a uma solução ser proposta pelo sistema RBC, o desempenho dessa solução pode ser medido automaticamente. Em um caso contrário, essa avaliação teria de ser feita manualmente, em uma segunda etapa, pelo usuário, que poderia em seguida atualizar a base de casos. Porém, o sistema visa a um usuário leigo.

• tipo de dados : essa informação envolve características dos valores como, por exemplo, se contínuos ou discretos; seqüenciais ou aleatórios e qual seu tipo de representação .

• precisão esperada da resposta da rede.

tempo despendido no aprendizado.

• tipo de aprendizado : estático ou dinâmico.

A parte de RBC do sistema pode trabalhar não somente com casos que apresentem novos atributos suplementares, como também pode criar novos atributos a partir das regras adicionadas à rede. Essas regras, como já foi dito, implementarão a parte do conhecimento existente na base de casos que provém do especialista. Os atributos do caso trazem as informações do usuário. O quanto cada parte irá crescer, regras e atributos do caso, será determinado na fase de avaliação do sistema. Uma quantidade muito grande de atributos necessários e fixos aumenta a rigidez do sistema, complicando a representação e aquisição de conhecimentos, enquanto facilita a escolha de uma solução adequada; um conjunto maior de regras de adaptação e pré-processamento aumenta o poder de generalização do sistema, mas exige uma análise mais cuidadosa por parte do projetista.

A definição da rede propriamente dita, a fim de que seja implementada no simulador, será fornecida pelos atributos da solução. O que vai orientar a escolha desses atributos serão os parâmetros requisitados pelo simulador (Neuro Solutions) para a construção do arquivo de descrição da rede. Também como uma primeira abordagem, a idéia é que sejam estipulados valores *default* para os parâmetros que não forem encontrados nas soluções. Idealmente, os atributos da solução estão em aberto, ou seja, não existe atributo obrigatório. Contudo, a fim de que o sistema seja coerente, alguns parâmetros devem constar da maioria das soluções: o modelo da rede, o algoritmo de aprendizado, número de camadas e a quantidade de neurônios por camada da rede. Outros parâmetros usuais, por exemplo, taxas de aprendizado ou eventuais taxas de *momentum*, podem ser aproveitadas para testes automáticos na fase de

avaliação. Pequenas variações em torno de um valor inicial permitiriam a criação de diversos subcasos com o objetivo de busca de uma solução ótima. Essas variações seriam regidas também por regras de reparo, adicionando, mais uma vez, uma parte de conhecimento advinda do especialista.

A conexão da saída do sistema RBC com o simulador pode ser realizada através de arquivos de *script*, já adequadamente formatados. Isso permite a formação de um ambiente único. Esse ambiente recebe as especificações do usuário e utiliza o modelo híbrido para gerar e testar uma solução adequada, retornando a resposta ao usuário. Um possível procedimento de avaliação seria a comparação dessas redes propostas com sugestões diretas de especialistas e com casos já documentados não incluídos na base.

## 5.3 Escolha de Atributos

Para o domínio de projeto de RNAs, os campos do caso foram escolhidos *a priori* a partir de características consideradas relevantes na escolha do modelo e que sejam existentes na maioria dos paradigmas. São formados por dois tipos básicos de informação: informações relacionadas ao conjunto de dados e informações relativas à aplicação desejada.

A sugestão inicial de atributos busca determinar o perfil estatístico do conjunto de dados. Uma vez que as primeiras RNAs escolhidas implementarão classificadores, esses atributos procuram caracterizar a maior ou menor facilidade que o modelo terá em agrupar as classes. Conjuntos de dados que possuam amostras consistentes (classes bem definidas) e bem equilibradas, indicam um aprendizado mais rápido e um desempenho melhor. Os parâmetros descritos pelos atributos de solução do caso refletem esse desempenho. Em outras palavras: o caso apresenta uma situação em que a implementação de um classificador foi bem sucedida quando aqueles determinados parâmetros foram usados associados a um conjunto de dados que apresentava o perfil estatístico descrito.

Existe uma grande quantidade de índices estatísticos que poderiam ser empregados. Porém, a presença de uma quantidade elevada de grandezas pedidas durante a consulta à base, grandezas essas que devem ser fornecidas pelo usuário, contribui muito para uma diminuição na usabilidade do sistema. Partindo da premissa de manter esse conjunto de índices o menor possível, foram escolhidos como representativos: a quantidade de padrões, o índice de assimetria total do conjunto e o índice de achatamento (*kurtosis*) de cada amostra. Esses índices, que serão melhor explicados na próxima seção, fazem parte de um conjunto maior de índices estatísticos adotado pelo Projeto StatLog [Michie et al. 1994] para análise de classificadores. Naquela ocasião diversos tipos de abordagens foram considerados. Nesta tese,

por questões de simplificação de cálculo e maior pertinência com o comportamento de abordagens conexionistas, foi escolhido esse conjunto mais reduzido.

Esses valores são calculados diretamente no caso de atributos numéricos contínuos. No tratamento de atributos simbólicos (enumerações, por exemplo), que vêm a ser representados geralmente por valores numéricos discretos, não faz sentido o cálculo de todos os quatro índices. Duas alternativas podem ser adotadas: ignorar os atributos não numéricos (ou eventuais numéricos discretos) e incluir como atributo a proporção entre estes e os não simbólicos ou associar cada um dos valores estatísticos escolhidos a um correspondente análogo discreto. Por exemplo, o grau de achatamento de uma distribuição discreta pode ser obtido pela análise de seu desvio-padrão em conjunto com o cálculo de determinados centis; um índice de assimetria pode ser calculado pela análise dos valores de moda e mediana. As duas alternativas podem ser comparadas, mas apenas a primeira foi empregada nesta tese.

Finalmente, as informações relativas à aplicação especificam melhor a operação desejada. O tipo de aplicação é um atributo que geralmente será de grande peso para a escolha do modelo de RNA. Na implementação desta tese, foram incluídos apenas classificadores, mas é prevista uma expansão para RNAs que executem clusterizações, regressões e otimizações também. O número de classes e a quantidade de características consideradas também integram o conjunto de atributos do caso.

### 5.3.1 Índices Estatísticos

É importante ressaltar que a presença de índices estatísticos nos campos de definição dos casos não pretende estabelecer uma relação causa/efeito determinística entre o perfil do conjunto e os parâmetros finais da rede. O objetivo é fornecer características ao caso que sirvam apenas de indícios úteis durante o processo de associação. Conforme já foi afirmado, não é esperado dentro desse tipo de domínio um comportamento que possa ser claramente expresso por regras, por exemplo. Desta forma, um determinado perfil estatístico nem sempre será adequadamente tratado por uma RNA modelada com uma certa combinação de parâmetros, porém certamente este perfil é útil na condução do processo iterativo de construção da RNA, oferecendo ao menos um ponto inicial não muito distante do ponto ótimo.

Três índices que expressam o perfil estatístico do conjunto de dados foram levados em consideração. O primeiro e mais simples deles é o número de padrões existentes, ou seja, de quantas amostras de padrões o conjunto de dados dispõe. Conjuntos pequenos podem não possuir um número de amostras expressivo, conjuntos muito extensos podem exigir um treinamento mais demorado, por exemplo.

Os outros dois índices procuram expressar o quanto as amostras dos padrões contidas no conjunto de dados para cada classe desviam-se de uma distribuição normal. Em princípio, é tão mais fácil para uma RNA classificar corretamente padrões quanto eles não se encontrem caoticamente espalhados. A idéia é que a existência de uma distribuição normal em torno de um vetor médio de cada classe seja uma característica relevante. A reunião final de classes normalmente distribuídas tende a criar uma distribuição uniforme do conjunto de dados. Uma distribuição uniforme não cria tendências internas no treinamento da RNA, sugerindo um ciclo de aprendizado mais curto.

O primeiro índice que expressa o desvio de um conjunto de dados é o índice de assimetria (A), ou coeficiente do momento de assimetria. É calculado pela fórmula abaixo:

$$A = \frac{m_3}{\sigma^2} \qquad (5.1)$$

O termo $m_r$ é o momento de ordem $r$, calculado por:

$$m_r = \frac{1}{N}\sum_{i-1}^{N}\left(x_i - \bar{x}\right)^r \qquad (5.2)$$

onde:

$N$ é a quantidade de padrões;

$x_i$ é o valor do atributo de ordem $i$;

$\bar{x}$ é o valor da média aritmética dos valores do atributo x, calculada por:

$$\bar{x} = \frac{1}{N}\sum_{i-1}^{N} x_i \qquad (5.3)$$

O termo $\sigma^2$ é a variância, calculada por:

$$\sigma^2 = \frac{\sum_{i-1}^{N}\left(x_i - \bar{x}\right)^2}{N} \qquad (5.4)$$

Este índice mede o grau de desvio ou afastamento do eixo de simetria de uma distribuição. Para distribuições assimétricas a média tende a situar-se do lado da cauda mais longa da distribuição.

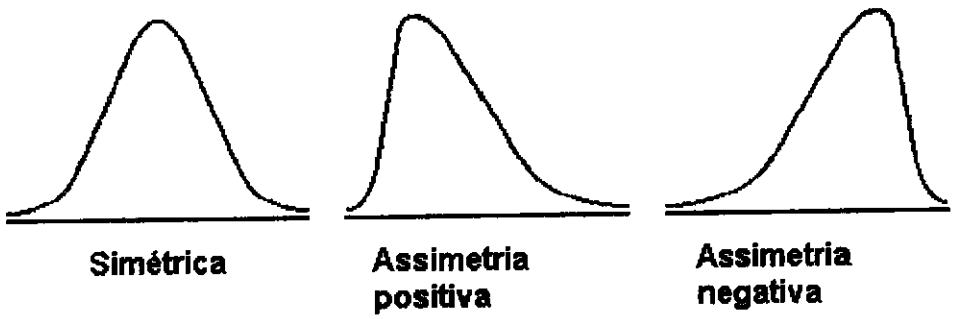


Figura 5.1 – Perfis de distribuições quanto à assimetria

O outro índice é conhecido como coeficiente de *Kurtosis* (*K*), ou coeficiente de achatamento. Seu valor é obtido pela fórmula abaixo :

$$K = \frac{m_4}{\sigma^4} \qquad (5.5)$$

O termo $m_4$ é calculado também pela fórmula 5.2 e o termo $\sigma^4$ é o quadrado da variância, obtida a partir da fórmula 5.4.

O coeficiente de *Kurtosis* indica o quanto uma distribuição foi achatada, isto é, o quanto ela aproxima-se de uma distribuição uniforme, ou se afasta de um perfil Gaussiano (normal).

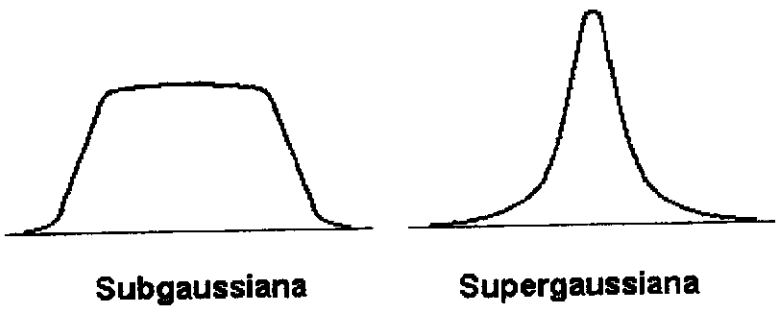


Figura 5.2 - Perfis de distribuição quanto ao achatamento

Nem toda distribuição simétrica é normal, mas a distribuição normal é simétrica. Da mesma forma, nem toda distribuição com baixo índice de achatamento é normal, mas esta é uma característica da distribuição normal. A combinação dos dois valores para um mesmo conjunto de dados é que pode mostrar sua aproximação da normal. Uma distribuição normal possui coeficiente de assimetria igual a zero ($A = 0$) e seu coeficiente de *Kurtosis* é igual a três ($K = 3$).

Esses coeficientes, na verdade são medidos para cada valor de atributo contínuo dentro de uma mesma classe. Em seguida é calculada a média aritmética dos valores dos atributos, o que fornece coeficientes médios de assimetria e *Kurtosis*. Este cálculo expressa a normalidade da classe. A distribuição total da amostra do conjunto de dados será tão mais próxima da normal quanto for composta de distribuições de classe normais. Os índices finais são médias globais das classes e, dentro do escopo específico desta tese, não tem seu valor submetido a uma análise estatística. Este valor serve, como já foi explicado, como uma característica do conjunto de dados. Sua interpretação não vem ao caso, embora seja esperado que expressem um perfil característico útil para o ciclo RBC.

### 5.3.2 Atributos Índice

Dentre os atributos do caso, um subconjunto, geralmente formado por um ou dois atributos, assinala os atributos índice. Estes atributos devem estar presentes em todos os casos da base e têm por finalidade orientar e simplificar a busca durante a fase de indexação. São escolhidos a partir de características decisivas do ponto de vista do projetista. Os índices escolhidos para as RNAs são o tipo de aplicação (no momento, sempre classificação) e o tipo dos valores ( numéricos e simbólicos).

### 5.3.3 Organização da Memória

É possível organizar a memória de um sistema RBC basicamente de duas formas: organizar a base de casos segundo uma estrutura hierárquica definida ou mantê-la sem nenhuma distinção de prioridade.

A primeira maneira exige um maior investimento durante a fase de aquisição do conhecimento, já que na verdade é introduzida maior quantidade de informação na base. Essa informação irá acelerar o processo de recuperação, facilitando a indexação de um caso. Memórias organizadas hierarquicamente aumentam o desempenho final do sistema, por conseguinte.

A segunda alternativa, adotada neste trabalho, é não criar nenhuma distinção entre os casos da base, construindo o que se costuma chamar de "memória *flat*". Os casos são armazenados seqüencialmente e o processo de recuperação executa uma busca simples. Ainda que esta decisão não contribua para uma melhoria no desempenho do sistema, ela facilita especificamente a análise de desempenho de um método de recuperação conexionista como o que é proposto nesta tese. Para que seja analisada a capacidade associativa do mecanismo indexador/adaptador empregado é mais conveniente que a base seja seqüencialmente

apresentada à RNA, sem ser necessário representar qualquer distinção de casos ou formar uma hierarquia de índices. Esta razão e a simplicidade de implementação fizeram com que o modelo de memória *flat* fosse adotado na implementação do RaBeCa.

## 5.4 Origem dos Casos

Para a verificação do comportamento do modelo baseado em RBC, foi construída uma base de casos formada por instâncias de RNAs que utilizam paradigmas básicos e que foram aplicadas a tarefas distintas. Essa base foi posteriormente acrescida das regras de transformação que foram julgadas necessárias para o ajuste das redes. Os casos são provenientes das soluções apresentadas por alunos da disciplina de Redes Neurais ministrada em nível de pós-graduação no ICMC, dissertações de Mestrado previamente defendidas também no ICMC e artigos técnicos publicados. Estes artigos foram extraídos de Anais de congressos envolvendo RNAs, publicados como *Lecture Notes* pela Springer Verlag [Orr, Müller 1998], de redes apresentadas em conferências, como no NIPS – *Advances in Neural Information Processing Systems* [Jordan et al. 1998] e de redes estudadas durante o projeto StatLog [Michie et al. 1994]. Foram escolhidos problemas que consistiam de um conjunto de dados fornecido, ou de acesso público, e uma tarefa proposta para ser executada através de um determinado modelo de RNA.

No desenvolvimento desta tese foram cobertos 3 modelos (redes MLP, mapas de Kohonen, e redes RBF) [Haykin 1999], que envolvem sempre atividades de classificação. Para os casos em que se dispõe do conjunto de dados, eles foram obtidos a partir de arquivos disponíveis em repositórios justamente para esse fim (arquivos de dados do projeto Statlog, repositório de Aprendizado de Máquinas da Universidade da Califórnia, Irvine) ou fornecidos pelo próprio desenvolvedor da rede.

A essas soluções foram adicionados casos considerados prototípicos de cada modelo, aplicados a outros conjuntos de dados não tratados. Estes últimos casos foram criados diretamente por um especialista em RNAs. Dessa forma, espera-se criar uma base rica o suficiente em informações para que a relação entre as regras de adaptação e a variedade dos casos da base possa ser avaliada. Essa relação deverá ser tal que o sistema não seja capaz apenas de escolher guiado pelas regras, isto é, afastar-se em demasia do ponto inicial escolhido pela similaridade de casos. A experimentação a ser realizada nessa etapa será justamente destinada a determinar uma proporção entre os atributos do caso e as regras de adaptação. Os atributos serão responsáveis pela exploração do espaço de soluções, de um modo global, enquanto as regras procuram localmente exaurir as possibilidades.

## 5.5 Generalizações

As generalizações permitem que seja realizado um cálculo de similaridade mais sofisticado. Elas permitem tanto que atributos simbólicos distintos sejam agrupados sob um mesmo nome e considerados similares como que atributos numéricos sejam reunidos em intervalos e vistos como iguais. Apenas este segundo tipo de generalização (numérica) foi usado, para estabelecer faixas similares dos índices estatísticos do tamanho da amostra.

O problema a ser resolvido por uma RNA, proposto pelo usuário, não está sendo pré-processado, inicialmente. Ainda que as regras de pré-processamento por vezes sejam disparadas, elas são usadas apenas para execução de programas destinados ao cálculo de valores estatísticos não conhecidos. Regularmente, apenas a regras de reparo estão sendo empregadas.

Essas regras manipulam a topologia da rede, estabelecendo um número de unidades (neurônios) adequado em cada camada. Esse tipo de conhecimento é muito mais facilmente incluído desta forma do que se tentasse obtê-lo diretamente da generalização dos casos.

## 5.6 Os Casos

Esta seção ilustra o ambiente RaBeCa, mostrando seus *scripts* e telas, de forma a fornecer um exemplo conciso da representação da base de casos descrita nas seções anteriores.

A Figura 5.3 mostra os campos de um caso segundo a escolha de atributos descrita na seção 5.1.

```
case definition is
        field Descrição_dos_Dados type is string weight is 0;
        field Natureza_do_Problema type is
                (classificação, regressão, extração_de_características) weight is 2;
        field Tipo_dos_Valores type is
                (numéricos, simbólicos_não_ordenados, simbólicos_ordenados) weight is 0;
        field total_de_amostras type is number weight is 2;
        field exemplares_treinamento type is number ;
        field exemplares_teste type is number;
        field Qtd_Classes type is number weight is 2;
        field Qtd_Atributos type is number weight is 2;
        field kurtosis type is number;
        field assimetria type is number;
        field conjunto_de_dados type is string;
end;
```

Figura 5.3 -Campos de um caso de RNAs

A tela gerada pelo RaBeCa após a interpretação do script descritor da base é mostrada na Figura 5.4, já com valores escolhidos pelo usuário.

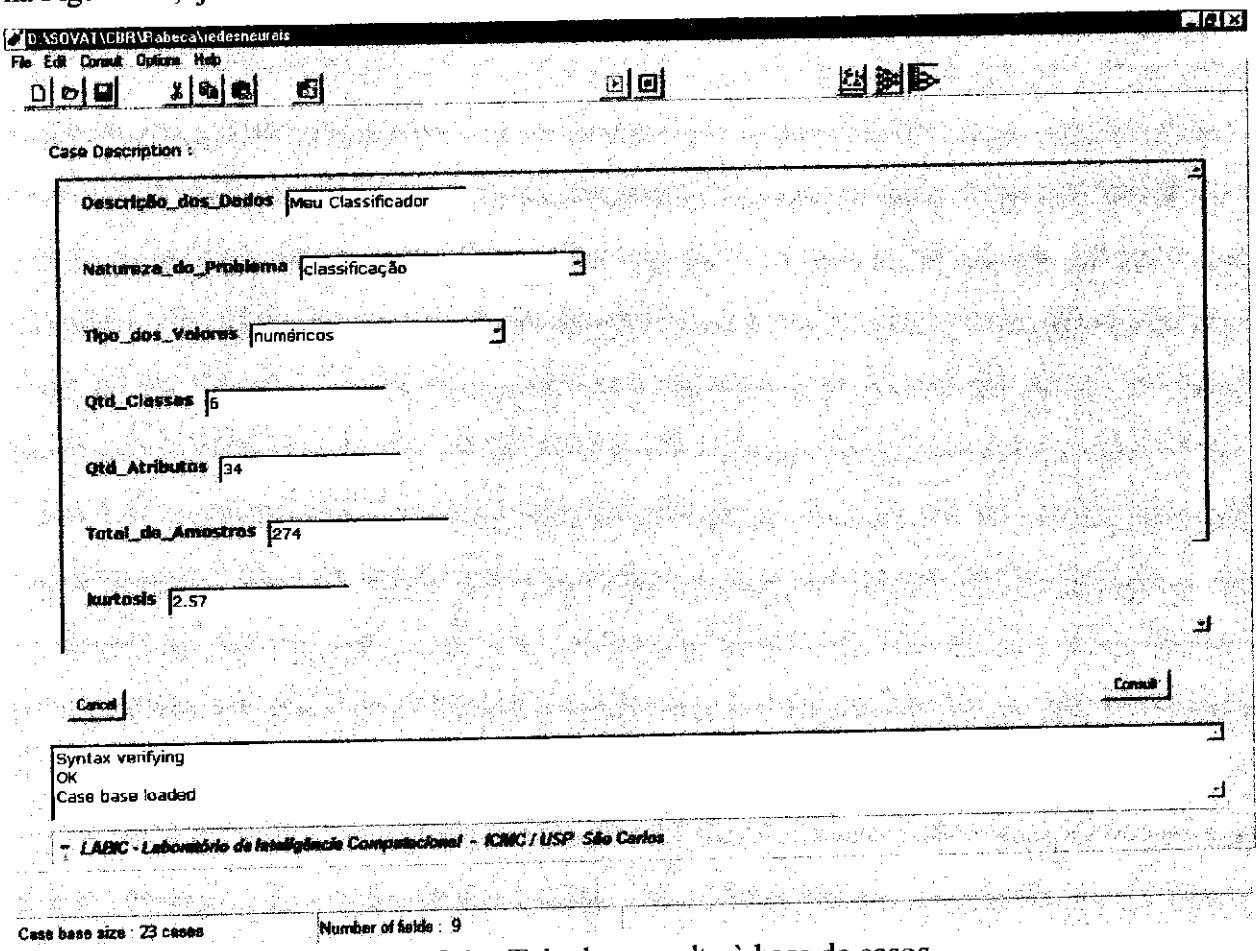


Figura 5.4 – Tela de consulta à base de casos

A escolha dos atributos índice é mostrada na Figura 5.5.

```
index definition is

        index on Natureza_do_Problema;
        index on Tipo_dos_valores;

end;
```

Figura 5.5 - Atributos índice

Os atributos criados a partir das definições dos campos do caso são mostrados na Figura 5.6. Os atributos índice estão marcados por um asterisco.

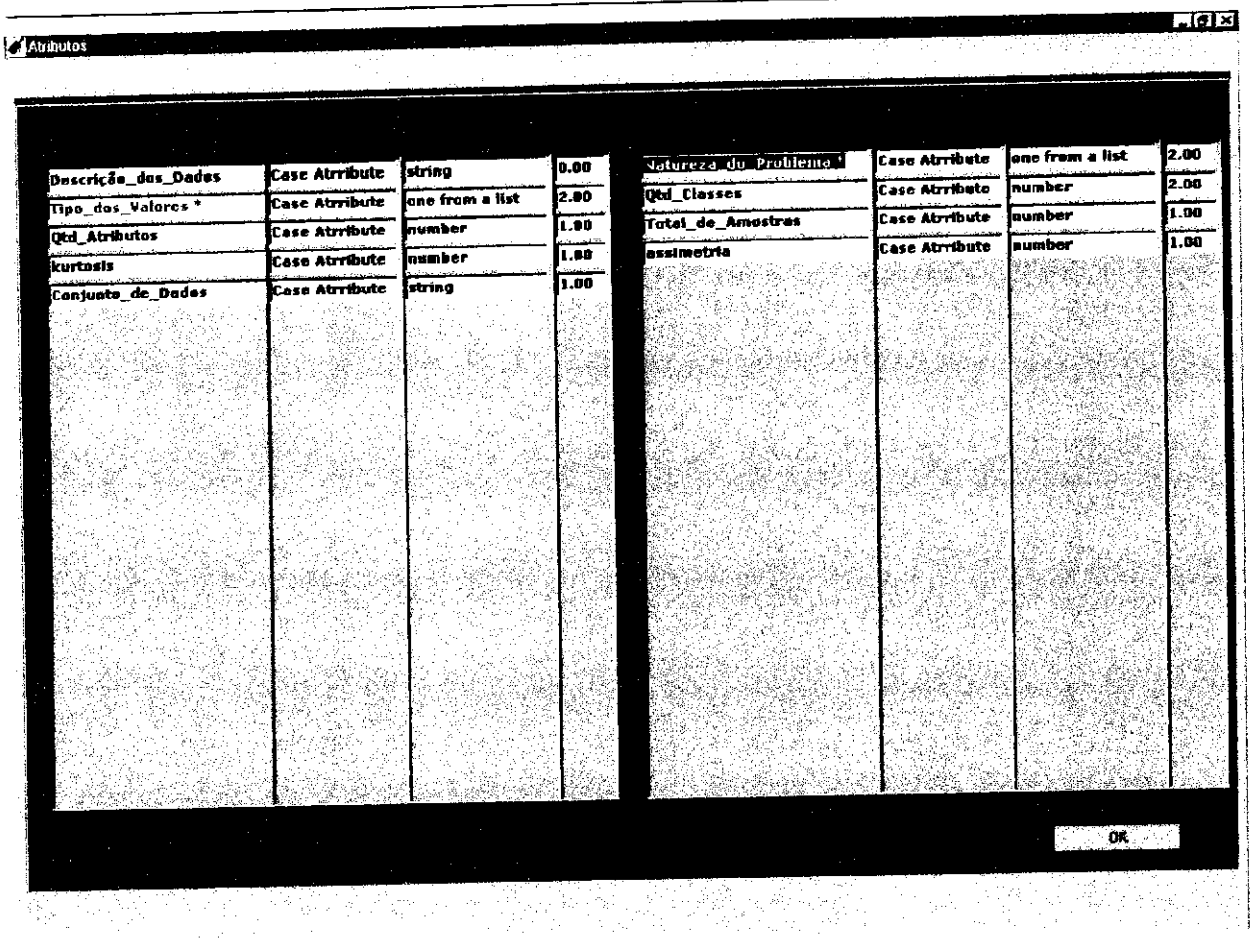


Figura 5.6 - Atributos do Caso

Em seguida, são criadas regras de pré-processamento a fim de adequar o caso do usuário a restrições gerais já conhecidas de antemão e que valem para qualquer caso proposto. No exemplo da Figura 5.7, duas regras simples executam um programa externo para fornecer valores que permitam instanciar atributos não definidos pelo usuário. A Figura 5.8 mostra duas regras de reparo: uma para estabelecer a topologia da RNA em relação à quantidade de neurônios por camada e outra para escolher o tipo de validação executada, em função do tamanho da amostra disponível. Finalmente, a Figura 5.9 mostra a janela de regras do RaBeCa, especificamente as regras de reparo.

```
preprocess rule definition is

repair rule testa_exec is
        when
            kurtosis is undefined
        then
            execute "CalcKurt.exe";
end;

repair rule testa_exec is
        when
                assimetria is undefined
        then
                execute "CalcAssim.exe";
        end;

end;
```

Figura 5.7 - Regras de Pré-processamento

```
repair rule definition is

        repair rule ajusta_topologia is
        when
                1 = 1 ~ sempre executa
        then
            evaluate neurônios_de_entrada to qtd_atributos;
            evaluate neurônios_de_saida to qtd_classes;
            evaluate tam_camada_escondida to log2(Total_de_amostras) + 1;
        end;

        repair rule tipo_validacao is
        when
                Total_de_amostras < 1000
        then
                change partição to "cv_10_fold";
        end;
end;
```

Figura 5.8 - Regras de reparo

As generalizações referenciadas na seção 5.3 são estabelecidas na seção de modificações, segundo a sintaxe da CASL. Como pode ser visto na Figura 5.10, generalizações definidas pela palavra **abstraction** criam generalizações a partir de atributos enumerados. Neste caso, incluindo sob "regressão" tipos de problemas definidos como "previsão" ou "ajuste" e igualando os termos "*clustering*" ou "agrupamento" a "clusterização". O termo **similar range**, por sua vez, torna similares valores numéricos contidos dentro do intervalo que ele define. Também neste caso, o tamanho da amostra, ou seja, a quantidade de padrões, é agrupada em faixas, que podem ser superpostas ou não. O resultado visual deste trecho de *script* dentro do RaBeCa é mostrado nas Figuras 5.11 e 5.12.

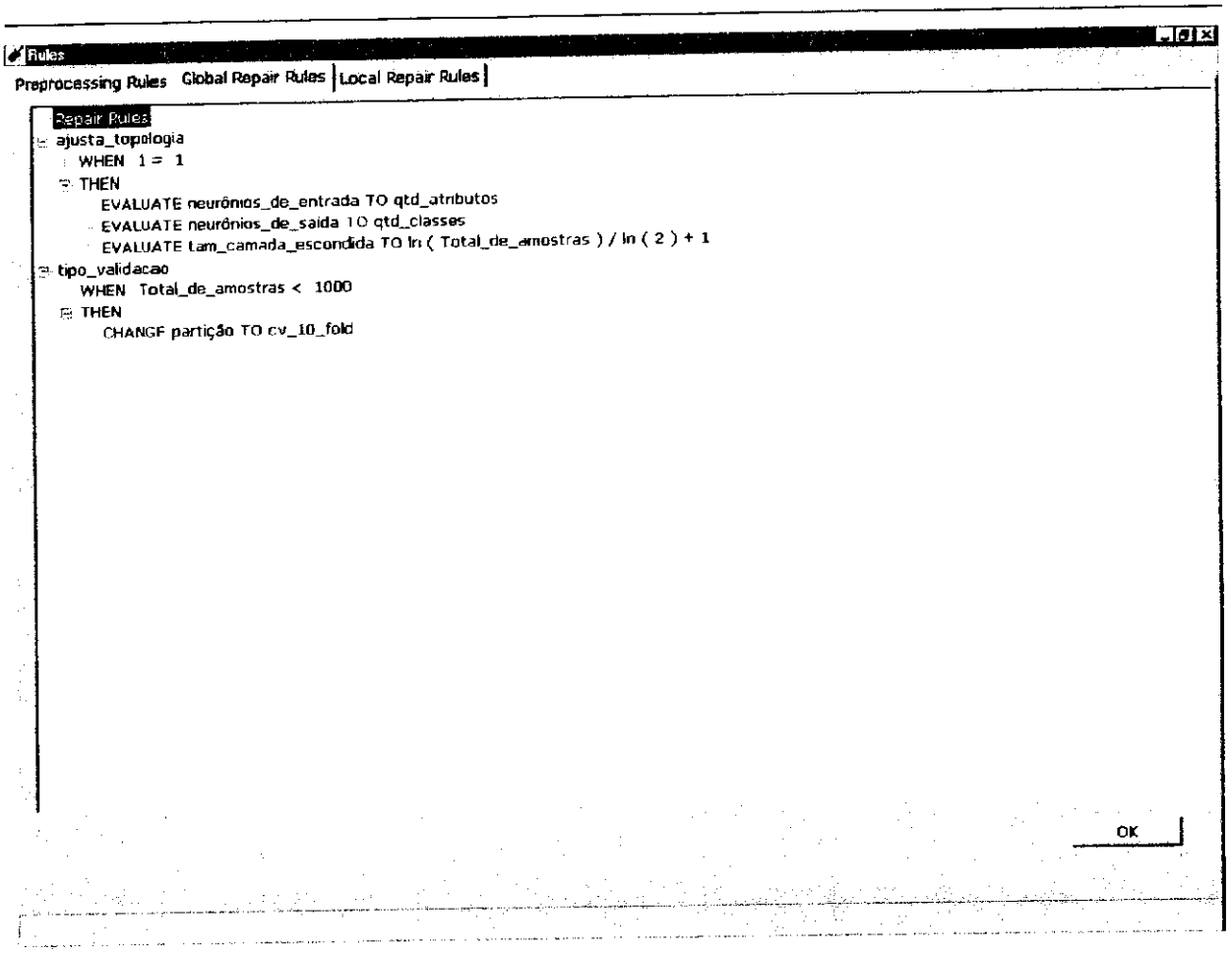


Figura 5.9 - As regras exibidas dentro do RaBeCa

```
modification definition is

        abstraction regressão is (previsão, ajuste);

        abstraction clusterização is (clustering, agrupamento);


        field Total_de_amostras similar range 100 to 500;

        field Total_de_amostras similar range 500 to 1000;

        field Total_de_amostras similar range 1000 to 5000;

        field Total_de_amostras similar range 5000 to 20000;

end;
```

Figura 5.10 - Generalizações empregadas

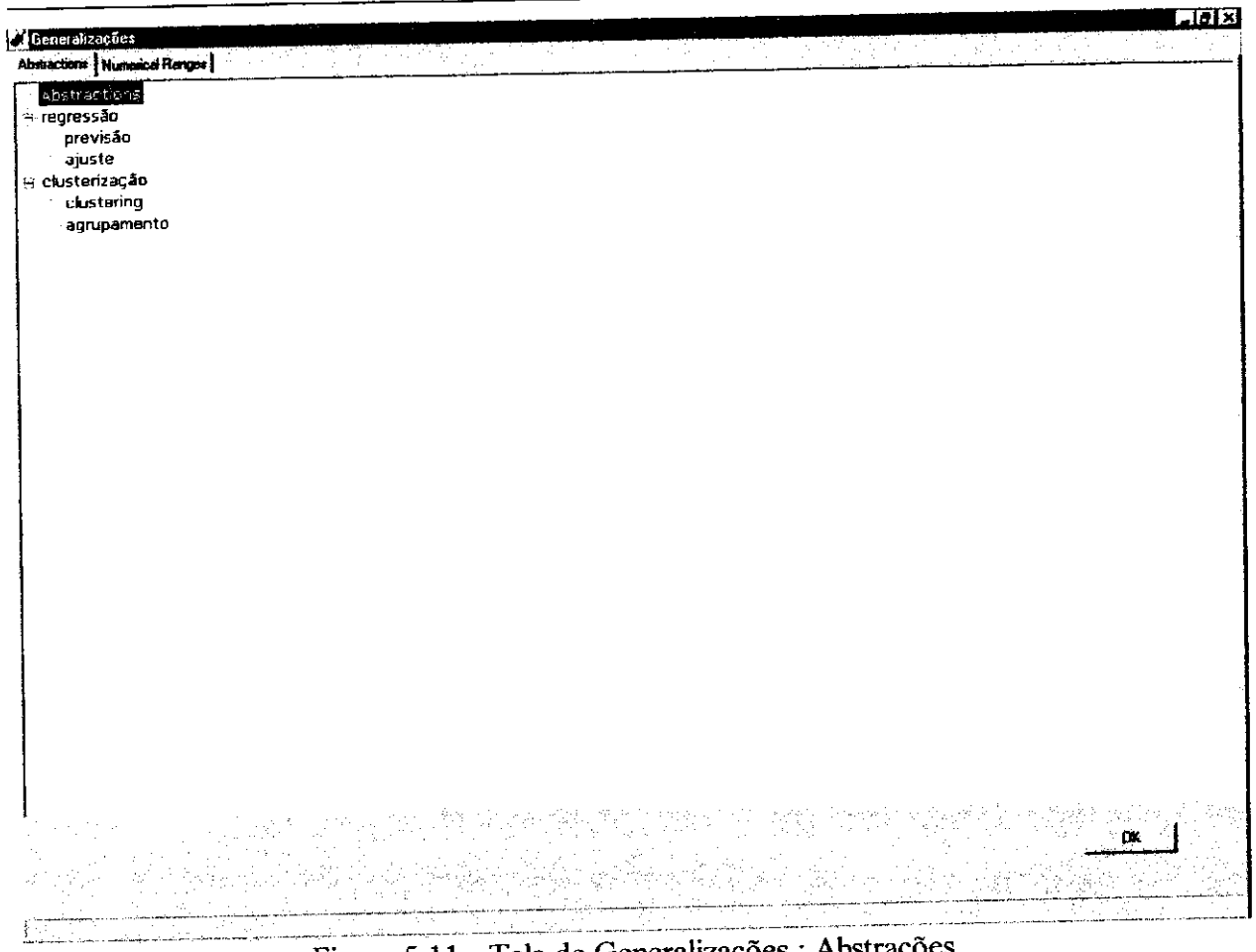


Figura 5.11 - Tela de Generalizações : Abstrações

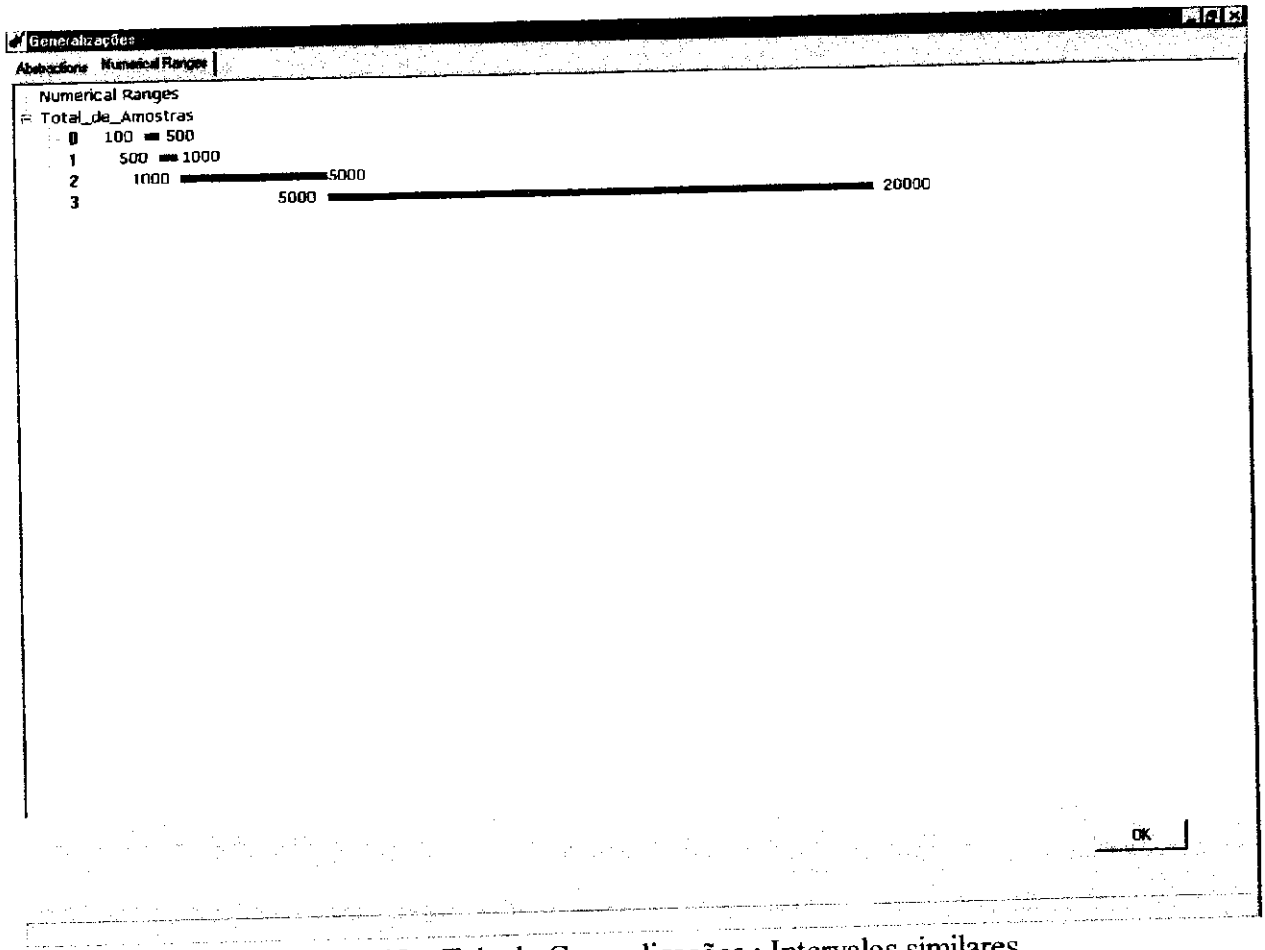


Figura 5.12 - Tela de Generalizações : Intervalos similares

Até este ponto, foi mostrado como o ambiente RaBeCa reflete as cinco primeiras seções do *script* descritor da base de casos. A fim de apresentar um caso em si, com seus atributos e valores, é mais prático mostrar a tela exibida após ser efetuada uma consulta à base de casos, em seguida à tela mostrada na Figura 5.4. A partir dos valores introduzidos pelo usuário e mostrados naquela figura, o RaBeCa irá apresentar o primeiro caso do ranking de casos recuperados, começando pelo de maior similaridade. A exibição do caso em tela é mostrada na Figura 5.13. A descrição do caso como se encontra na base está na Figura 5.14.

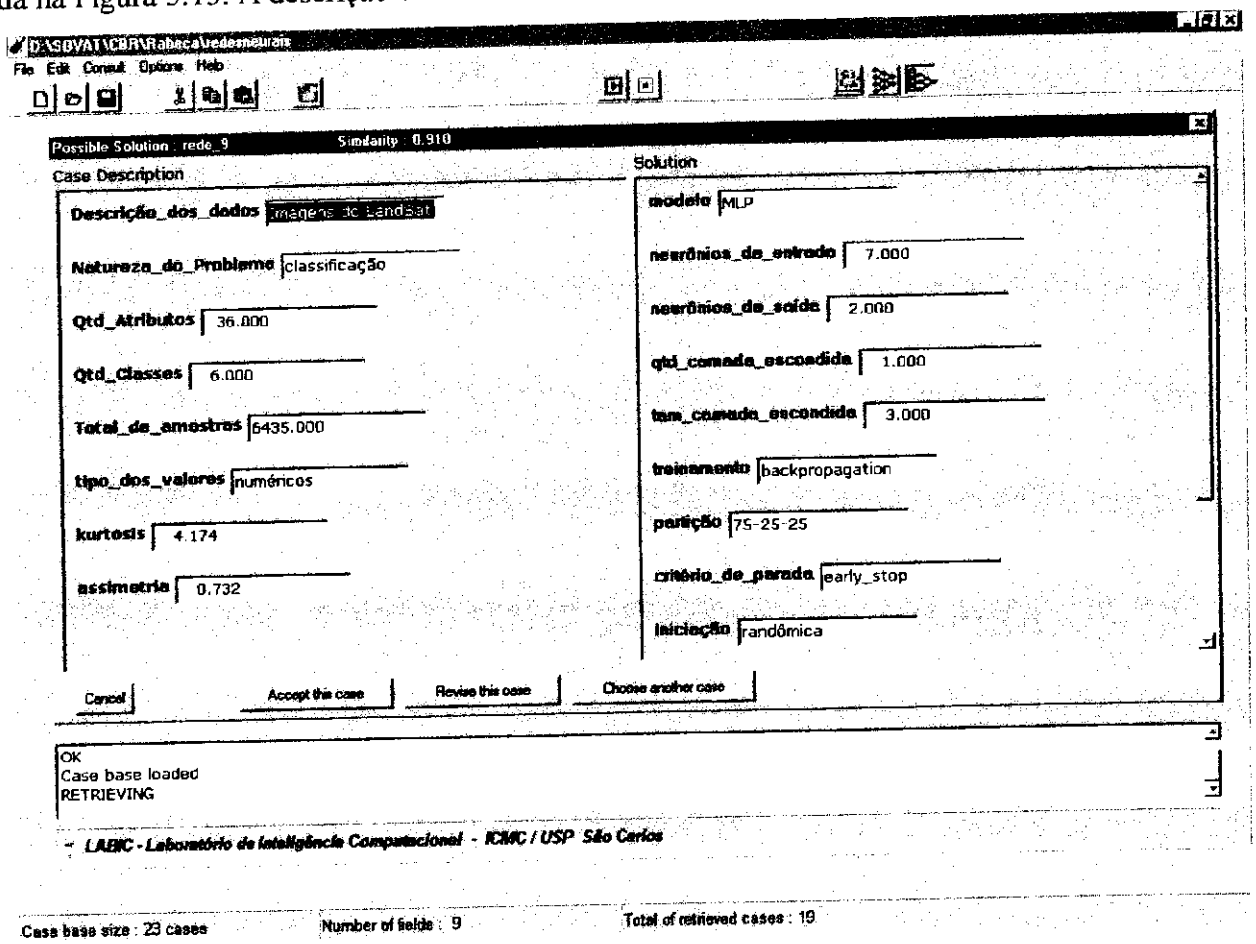


Figura 5.13 - Caso Recuperado

Os passos anteriores descrevem a introdução dos casos da base de modelos de RNAs na memória do sistema, o estabelecimento de seus campos e índices e as transformações e generalizações que podem sofrer. O próximo passo nesta seqüência é a etapa de adaptação. Ela pode ser realizada manualmente na tela mostrada na Figura 5.13 ou através das regras de reparo, pela escolha do botão correspondente. O caso novo será resultante da troca dos atributos de caso do caso recuperado pelos atributos de caso fornecidos e pelos atributos de caso e de solução transformados pelas regras de reparo. Este caso novo é exibido na tela no mesmo formato do caso recuperado, podendo ser editado manualmente, aceito ou descartado. A Figura 5.15 mostra esta janela de revisão. Neste ponto, o objetivo é descrever a estrutura e a

representação da base de casos dentro do ambiente. No capítulo 7 serão discutidos demais aspectos de uma consulta à base, através de um estudo de caso.

```
case instance rede_9 is
        Descrição_dos_dados = 'Imagens do LandSat';
        Natureza_do_Problema = classificação;
        Qtd_Atributos = 36;
        Qtd_Classes = 6;
        Total_de_amostras = 6435;
        tipo_dos_valores = numéricos ;
        kurtosis = 4.1737;
        assimetria = 0.7316;

solution is
        modelo = MLP;
        neurônios_de_entrada = 7;
        neurônios_de_saída = 2;
        qtd_camada_escondida = 1;
        tam_camada_escondida = 3;
        treinamento = backpropagation;
        partição = '75-25-25';
        critério_de_parada = early_stop;
        iniciação = randômica;
        ativação = sigmóide;
        atualização = síncrona;
        taxa_aprendizado = 0.4;
        momento = 0.1;
end;
```

Figura 5.14 - Descrição do caso na base

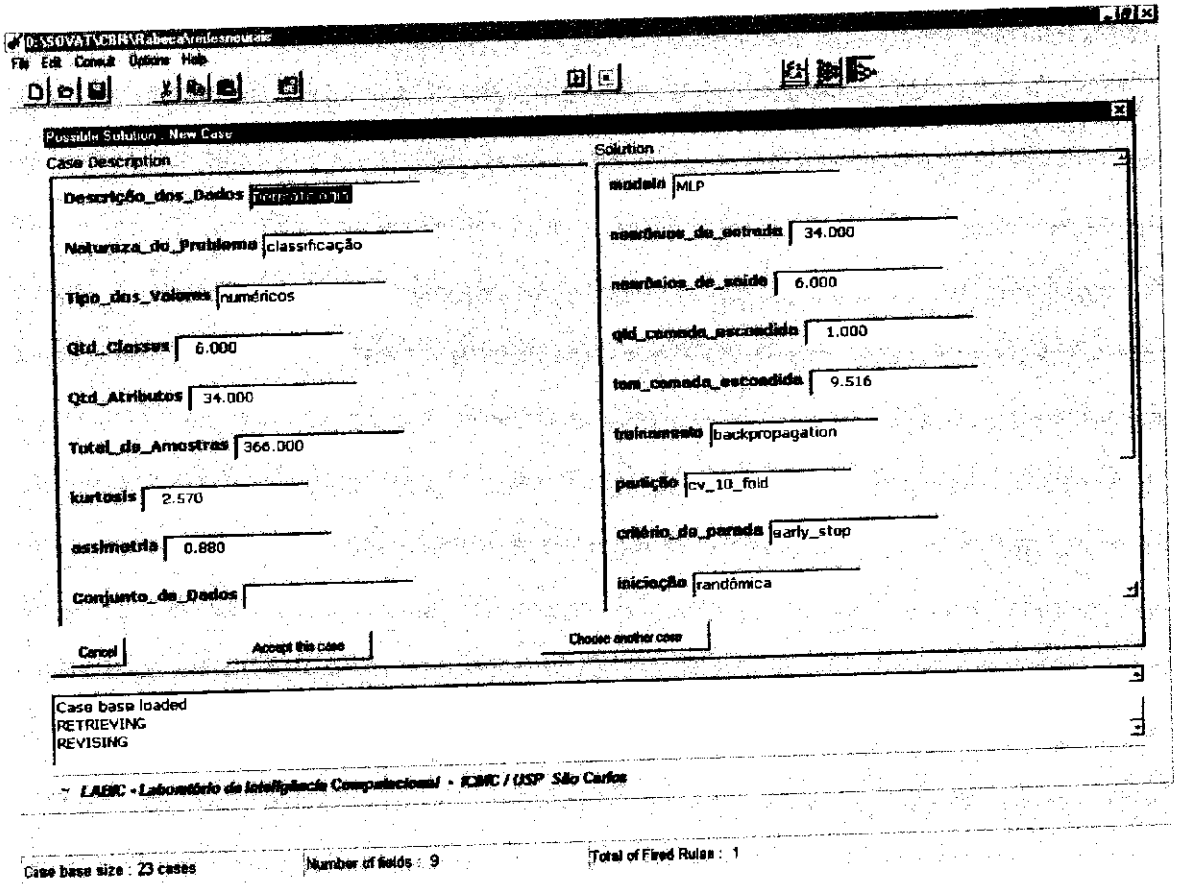


Figura 5.15 - Tela da etapa de revisão

## 5.7 Considerações Finais

A base de casos mostrada neste capítulo deve ser encarada como possuindo uma característica essencialmente dinâmica. Assim como o RaBeCa pretende ser uma ferramenta para experimentações, também esta representação adotada para a base de casos é apenas uma primeira instância de representação possível para o problema de escolha e configuração de uma RNA. Procurou-se mostrar o potencial de aplicação da metodologia e do programa desenvolvido no tratamento deste domínio. Outros atributos, regras e generalizações, sem dúvida, são possíveis. Porém, o conjunto aqui apresentado pretende ser um ponto de partida bastante conveniente. No próximo Capítulo, será abordada a questão de como é possível avaliar o desempenho de um sistema construído segundo estas premissas.

# Capítulo 6

# Metodologia de Avaliação

Para aproveitar uma característica muito interessante do problema de configuração de uma RNA, escolheu-se determinar parte da medida do desempenho das redes através de sua implementação em um simulador. Essa característica é a possibilidade de automatizar a etapa (do ciclo RBC) de reutilização de uma solução. A solução proposta pode mesmo vir a prescindir da análise de um usuário, tendo seu sucesso ou não determinado por uma execução na prática, dado que os arquivos de dados para teste estejam disponíveis.

Como escolha do simulador, foi empregado o Neuro Solutions. Este é um simulador bastante poderoso, que roda diretamente sob o MS-Windows. Apesar de seu código fonte não estar disponível para recompilação, pois ele não é um *software* gratuito, ele possui características muito interessantes para o projeto: é confiável, em virtude de sua larga utilização e recomendação por diversos usuários [Liu 1996], possui uma gama bastante grande de modelos implementados, gera código compilável das redes criadas e possui uma linguagem de *script* (macros) própria, o que, juntamente com a característica de ser compatível com o OLE do Windows (*Object Linking Embedding*) permite que sua execução seja controlada por outros programas sob esse mesmo sistema operacional. Já foi desenvolvido no Laboratório de Inteligência Computacional (LABIC) um simulador de RNAs, o Kipu [Vargas et al. 1997], que é executado em Windows e utiliza o mesmo formato de arquivo que o SNNS na descrição das redes, contudo, sua abordagem é mais focalizada em redes construtivas, fora do escopo deste trabalho.

Também a partir da base de casos construída, deseja-se verificar a eficiência do sistema. Essa verificação será realizada a partir da comparação do desempenho das redes sugeridas com as redes implementadas por um especialista para a solução do mesmo problema e com soluções escolhidas randomicamente, para que seja feita uma medição da distância que separa as soluções. Esses testes, mais do que avaliar o funcionamento do programa em si, destinam-se a

verificar a eficácia da abordagem, ou seja, se o desempenho de um usuário leigo durante a fase de aprendizado pode ser auxiliado por um sistema baseado em casos.

## 6.1 O Que Avaliar ?

Sendo este trabalho composto por dois grandes objetivos, ou seja, o estudo da inserção de uma RNA no ciclo RBC e a aplicação da metodologia RBC no problema de escolha de redes, uma avaliação destas propostas também necessita ser feita através de dois processos distintos.

A primeira etapa visa a verificar a viabilidade e a eficiência do sistema híbrido formado pelo ciclo RBC que executa os passos de recuperação e adaptação em uma só etapa, através da operação da memória associativa descrita. É necessário comparar o desempenho do RaBeCa em uma determinada base de casos. Primeiramente, usando apenas o algoritmo k-NN, e depois utilizando a RNA corrigida por este mesmo algoritmo. Preferencialmente, a base de casos envolvida deve possuir uma quantidade muito superior de casos à quantidade mínima necessária, para que a capacidade de generalização do indexador possa ser aferida. Ela também não deve ser a mesma construída para os testes da próxima fase, para que a aquisição de conhecimento do domínio não seja uma variável envolvida.

Além dessa avaliação, também deve ser medido o desempenho do sistema final, ou seja, do ciclo RBC (híbrido ou não) aplicado ao problema de escolha de redes. A seguir serão descritos os critérios escolhidos e a metodologia dos testes.

## 6.2 Critérios

Avaliar o desempenho de um sistema baseado em RBC ainda é uma tarefa que não dispõe de procedimentos bem estabelecidos. Porém, partindo de trabalhos publicados recentemente [Martins 2000], existem técnicas provenientes da área de Sistemas de Informação e de estudos de usabilidade. Não há ainda um procedimento padrão quanto a este ponto, mas é interessante que sejam levadas em consideração algumas possibilidades.

Para tal, estudou-se o emprego das seguintes métricas, a fim de verificar-se se são aplicáveis, (ou ao menos adaptáveis) ou não a RBC :

Métrica de precisão e Métrica de retorno

Consiste em relacionar o número de casos resgatados relevantes ao problema com o número de casos relevantes existentes (retorno) ou número de casos total resgatados (precisão).

O problema aqui é a determinação dessa relevância. Um caso relevante, em princípio, seria aquele que fosse resgatado por diversas consultas independentes. Isso provavelmente irá acontecer com os protótipos de cada rede. É uma característica do domínio escolhido. Por outro lado, como foi visto antes, uma vez que o número total de casos resgatados tende a ser baixo, o maior esforço será despendido na adaptação.

Métrica de refugo

Da mesma forma que o anterior, mas contando resgates de casos não relevantes. Talvez seja importante para determinar o ajuste do sistema na escolha do paradigma básico embora, mais uma vez, essa não será a principal prioridade do sistema.

Métrica de retorno de relevância única

Analogamente às anteriores, mas comparando diversos métodos de recuperação (resgate) de casos a fim de se verificar se um deles é o único capaz de resgatar um determinado caso relevante. Essas três primeiras métricas relacionam-se à fase de recuperação, não devendo ser de importância no domínio de RNAs.

Métrica de eficiência

Apesar de teoricamente relacionar-se ao custo de obter-se um determinado caso e, por causa disso, ser de difícil implementação, talvez possa ser obtida através de uma medição que relacione o desempenho da primeira rede sugerida e daquela finalmente aceita. Esse tipo de medida leva em consideração não só a fase de recuperação como as seguintes (reparo e reavaliação). Uma vez que se tem acesso ao funcionamento da ferramenta de RBC, podem também ser incluídos módulos de contagem de números de vezes que o usuário acessou uma determinada janela (configurando uma determinada fase) ou mesmo o sistema de ajuda. Esse tipo de métrica é mais objetiva e quantitativa. Ela pode ser aproveitada para determinar se o comportamento do espaço de soluções do problema de escolha do modelo conexionista é homogêneo ou não, analisando se realmente o usuário partiu de casos semelhantes mas obteve resultados muito diversos, ao menos nas primeiras iterações da fase de adaptação.

## 6.3 Sistema Híbrido

Entende-se aqui por sistema híbrido o grupo de procedimentos de recuperação e adaptação do ciclo RBC que utilizam a memória associativa proposta no capítulo 4.O objetivo da avaliação não é comparar o desempenho da indexação híbrida com o da convencional, isto é, não se deseja verificar se a recuperação de um caso é mais precisa através deste ou daquele método. A contribuição da proposta verifica-se no processo de adaptação, no qual é buscado um maior poder de generalização do algoritmo. Procura-se mostrar que o conhecimento

necessário à transformação do caso recuperado no caso que realmente atende ao usuário pode ser adquirido pela RNA após seu treinamento com os casos e soluções da base. Logo, a atenção deve ser concentrada na quantidade de regras de reparo que precisam disparar para a obtenção do par caso/solução final e no número de intervenções diretas do usuário através da interface. Quando da utilização da memória associativa, a única medida será a quantidade de intervenções manuais. Essa métrica procura indicar a dificuldade de transformação inerente a cada alternativa.

### 6.3.1 Metodologia de Avaliação do Sistema Híbrido

O processo de avaliação do sistema híbrido pode seguir os seguintes passos:

Estabelecer uma base de boa proporção (500 casos ou mais);

Treinar a memória associativa com estes casos;

Efetuar N recuperações e adaptações a partir de N casos do usuário usando o k-NN e regras apenas;

Efetuar as mesmas recuperações utilizando a memória associativa;

Medir a quantidade de casos diferentes recuperados, comparando-se o primeiro caso recuperado pelo k-NN e aquele recuperado pela memória associativa;

Para cada caso diferente, avaliar a qualidade do caso durante a fase de reutilização. Essa qualidade varia conforme o domínio da base, porém pode ser realizada por comparação com um subconjunto de casos não incluídos na base e de que se conhece uma boa solução. Uma vez que ambos os algoritmos retornem sua solução, ela é comparada com a que já se possuía, através de índices de similaridade. A média dos índices de similaridade é um índice de desempenho da memória associativa.

Aumentar o número de casos da base de casos progressivamente, repetindo os passos 3, 4 e 5. Quando a quantidade de casos diferentes ultrapassar uma porcentagem (determinada previamente) do valor anterior do passo 5, estabelecer o correspondente aumento do número de casos como sendo o limite acima do qual a memória associativa necessariamente precisa ser retreinada.

## 6.4 Sistema Final

Foi considerada como sistema final a união do ciclo RBC executado pelo RaBeCa com a execução do simulador de RNAs. A avaliação do sistema final, portanto, deve focalizar a eficiência da base de casos de redes neurais montada. Essa eficiência será traduzida tanto na facilidade de obtenção de um caso que seja considerado bom pelo usuário, como no

desempenho desse caso avaliado pelo simulador. Conforme mencionado anteriormente, o domínio de escolha de modelos de RNAs apresenta a possibilidade de automatizar parte da etapa de avaliação. Esta característica foi considerada importante neste trabalho, ao ponto de, diferentemente da avaliação do sistema híbrido, não serem levados em contas os passos necessários para a obtenção da solução em si, mas a qualidade dessa solução. Neste caso, então, não será empregada diretamente nenhuma métrica específica de sistemas RBC, mas sim deve ser levada em consideração o desempenho da RNA obtida após a consulta do usuário à base de casos.

## 6.5 Considerações Finais

O principal aspecto que deve ficar claro neste Capítulo é a existência de peculiaridades na avaliação de um sistema baseado em RBC. Apesar de algumas métricas apresentadas possuirem um caráter essencialmente numérico, a avaliação desejada nesta tese relaciona-se mais com aspectos de viabilidade da aplicação e da adequação ao usuário. Inicialmente, mais do que obter um índice de desempenho para a solução proposta, ou medir-se a capacidade da base de casos, deseja-se avaliar a proximidade, para o usuário, em termos de adaptações necessárias, entre sua primeira tentativa e soluções consideradas boas dentro de um domínio. À medida que a base de casos torne-se mais significativa, avaliações mais detalhadas serão possíveis. A metodologia de avaliação do sistema final será aplicada no Capítulo 7, em um estudo de caso.

# Capítulo 7

# Um Estudo de Caso: Classificação de Padrões

A fim de que fosse possível realizar uma avaliação da aplicação proposta nesta tese, foi realizado um estudo de caso a partir da base de modelos de redes neurais. A base de casos, montada conforme foi descrito no Capítulo 5, é composta dos casos obtidos do projeto StatLog e inclui principalmente soluções que utilizam o modelo MLP. Apesar de possuir um número reduzido de casos, 23, esses casos possuem a vantagem de terem sido selecionados por aquele projeto exatamente para a análise de problemas de classificação e apresentam uma descrição bastante precisa de seus atributos descritores (atributos do caso). Na maioria das vezes, os atributos de solução não estavam disponíveis em detalhes. Soluções incompletas não são um impedimento para o funcionamento do ciclo RBC, mas apenas dificultam a obtenção de uma avaliação precisa. Contudo, tendo em vista a necessidade de demonstrar o método proposto, sempre que foi encontrada uma referência ao uso de um determinado parâmetro sem que seu valor fosse fornecido, ele foi substituído por valores usuais, de modo a não distorcer em demasia a qualidade das soluções. A seguir será descrito o experimento imaginado e será realizada uma breve exibição e discussão dos resultados obtidos.

## 7.1 Descrição do Experimento

O primeiro passo após representar os casos do StatLog na sintaxe da linguagem CASL, foi determinar um conjunto mínimo de regras de pré-processamento e de reparo.

As regras de pré-processamento visam primeiramente a processar informações sobre as RNAs que já venham embutidas nas relações entre os atributos do caso. Para este domínio, a escolha mais imediata é empregá-las para instanciar, se possível, atributos ainda indeterminados relacionados aos índices estatísticos. Assim, estas regras verificam se o arquivo de dados está disponível e se os atributos de kurtosis e assimetria estão instanciados. Caso não estejam, um programa externo é chamado para extrair esse valor do conjunto de dados. Neste experimento

essas regras não foram disparadas, uma vez que os valores estatísticos já haviam sido calculados anteriormente.

As regras de reparo destinam-se a iniciar uma construção específica da rede a ser proposta pelo sistema. O primeiro ponto de aplicação é a determinação da topologia do sistema.

Uma vez que o objetivo é tratar problemas de classificação, é usual estipular-se o número de unidades da camada de entrada como sendo igual ao número de atributos e o número de unidades da camada de saída igual ao número de classes. Quanto ao número de unidades na camada intermediária, a opção foi trabalhar-se com uma heurística usual encontrada na literatura [Looney 1997] que estabelece este número segundo a equação 7.1 abaixo:

$$N_h = \log_2 (K) \qquad (7.1)$$

onde **K** é o número total de amostras do conjunto de treinamento e $\mathbf{N_h}$ é o número de unidades da camada oculta.

A segunda escolha de ponto de aplicação para as regras de reparo foi o tipo de partição usada para realizar-se a validação cruzada durante o treinamento da rede. O método de validação cruzada prevê uma divisão do conjunto de dados em duas partições, uma para treinamento e outra para validação. Periodicamente, durante a fase de treinamento, a RNA é testada com os padrões do conjunto de validação. A fim de manter-se uma boa capacidade de generalização, o treinamento é interrompido quando o erro médio quadrado calculado para a validação começa a crescer, mesmo que o erro calculado para o conjunto de treinamento continue a diminuir. Esse critério de parada é chamado de *early stop*. Usualmente, a partição usada para o treinamento é mantida em 50% dos padrões disponíveis, enquanto os outros 50% restantes são divididos em 25% para validação e, quando não existe arquivo específico para tal, 25% para teste. Essa é uma partição conhecida como 50-25-25. Outras proporções podem ser usadas. Porém, quando o número de amostras é considerado baixo (N < 1000, por exemplo), lança-se mão do expediente de repartir o conjunto de dados em n divisões, usualmente n=10, e ciclicamente usar 9 dessas partições para treinamento e uma para validação. Esse tipo de partição é conhecido como *n-fold*. A segunda regra de reparo estabelece este critério: quando o total de amostras é baixo, muda-se o tipo de partição.

Estabelecidas as regras de reparo e de pré-processamento, foram também determinados intervalos de similaridade para o tamanho da amostra e abstrações para a enumeração do tipo de problema. A partir deste ponto, o *script* descritor da base de casos ficou pronto e foi carregado no RaBeCa.

A métrica empregada para a avaliação deste experimento foi a métrica de eficiência descrita no Capítulo 6. Conforme foi explicado, essa métrica leva em consideração mais do que apenas a etapa de recuperação, já que vai medir a distância entre um caso adaptado e uma solução real. Para tal, foi usada uma solução real de um exercício proposto na disciplina de Redes Neurais ministrada em nível de pós-graduação no ICMC durante o segundo semestre de 2000. Neste exercício, foi pedido a 10 grupos de alunos que montassem e configurassem um classificador usando uma RNA usando modelo MLP e o aplicassem a um conjunto de dados contendo padrões de doenças dermatológicas. Esses padrões consistiam de vetores numéricos de 34 atributos cada. As doenças podiam ser de 6 tipos, ou seja, existiam 6 classes possíveis. O conjunto de dados possuía 366 vetores, todos rotulados, ou seja, associados a uma classe.

Os campos do caso definidos durante a construção da base de casos, descrita no Capítulo 5, foram então preenchidos de acordo com estes valores. Em seguida, foi realizada uma consulta à base de casos. O caso recuperado após ser examinado na tela do RaBeCa, foi submetido à revisão automática pelas regras, obtendo-se assim um caso novo. Este caso foi então comparado à solução mais comum entre as 10 apresentadas pelos alunos.

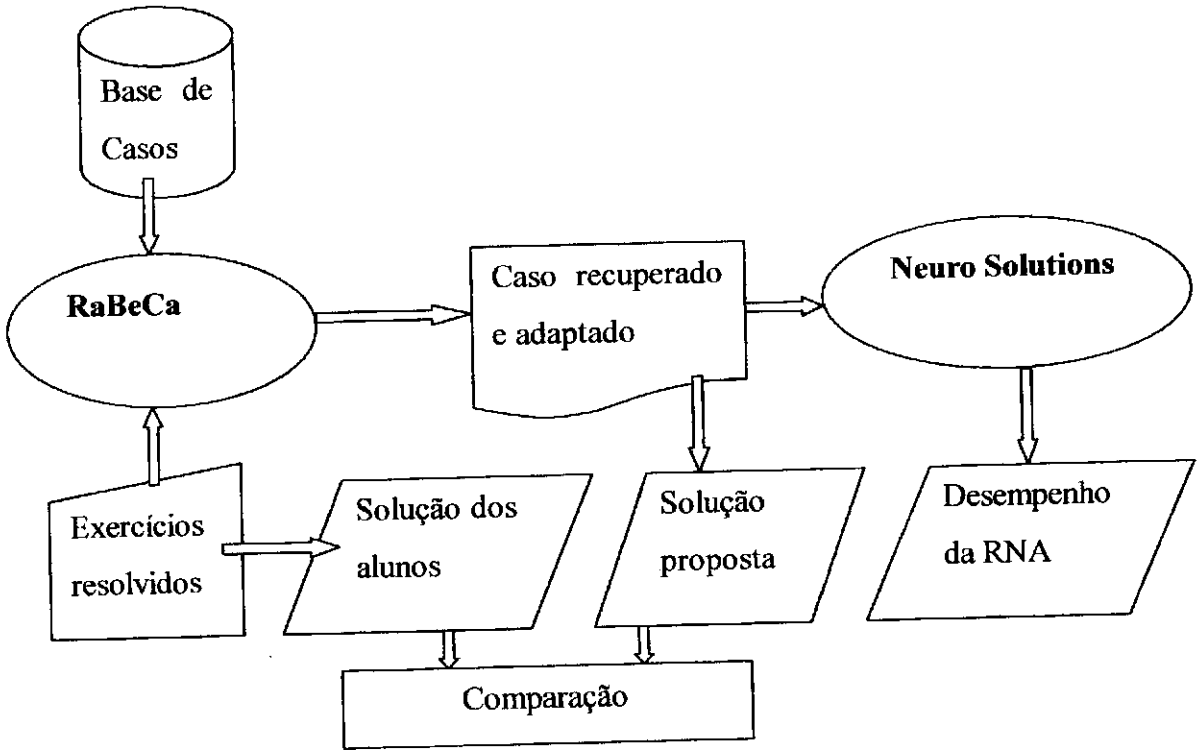


Figura 7.1 - Diagrama de blocos do experimento

Dessa forma, ainda que simples, pode-se mostrar o funcionamento da abordagem proposta, verificar as limitações do processo e medir a distância entre uma solução obtida por usuários não experientes e a solução do sistema.

O diagrama representando o experimento encontra-se na Figura 7.1. A tela do RaBeCa após a consulta à base pode ser vista na Figura 7.2. O caso adaptado é mostrado na Figura 7.3.

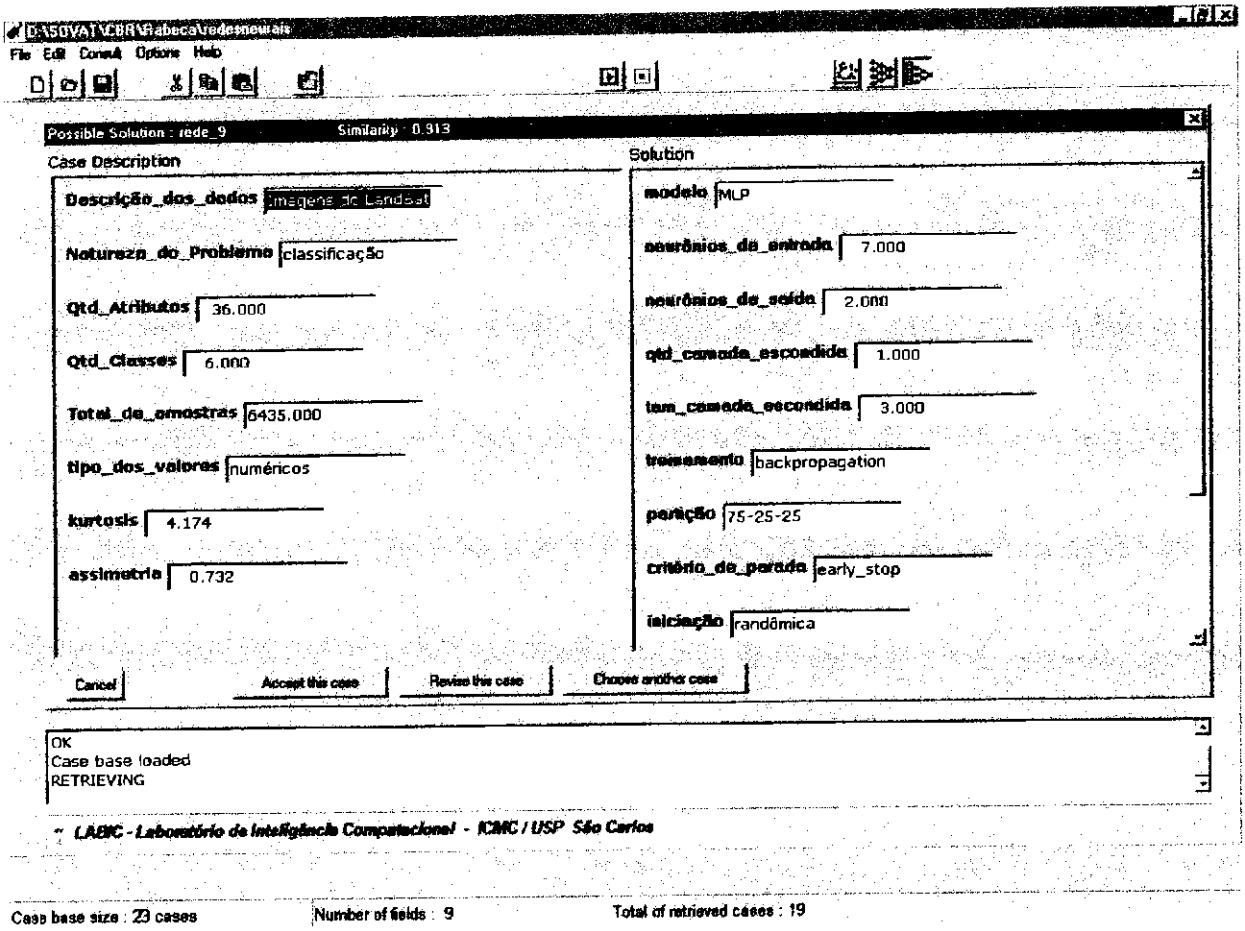


Figura 7.2 - Caso recuperado no experimento

## 7.2 Resultados

Nesta seção, são apresentados os valores dos atributos de solução de ambos os casos: o caso adaptado pelo sistema e o montado pelos grupos de alunos durante o exercício. Os resultados são comparados nas tabelas 7.1 e 7.2. A primeira coluna mostra os atributos das soluções; a segunda os valores da solução do caso adaptado; a terceira contém os valores da solução dos alunos e a quarta indica a diferença entre esses atributos, segundo o mesmo critério de similaridade descrito no item 4.1.2, no capítulo descrevendo o ambiente RBC. Desta forma, é como estivesse sendo medida a similaridade entre um caso novo e um não pertencente à base. O valor de similaridade final é a distância Euclidiana entre os dois vetores normalizada

pela distância mínima, isto é, se todos os índices fossem iguais a 1. Na Tabela 7.2 é mostrado o desempenho da rede original do primeiro caso recuperado, que tratava de outro conjunto de dados, da melhor rede dos alunos e da implementação da rede proposta pelo caso adaptado no NeuroSolutions.

| **Atributos Da Solução** | **Solução dos Alunos** | **Solução do Caso Adaptado** | **Similaridade** |
|---|---|---|---|
| Modelo | MLP | MLP | 1 |
| Neurônios de Entrada | 34 | 34 | 1 |
| Neurônios de Saída | 6 | 6 | 1 |
| Número de camadas escondidas | 1 | 1 | 1 |
| Tamanho da camada escondida | 15 | 9,52 | 0,56 |
| Treinamento | Backpropagation | Backpropagation | 1 |
| Partição | 50-25-25 | 10-fold | 0 |
| Critério de parada | Early stop | Early stop | 1 |
| Iniciação | Randômica | Randômica | 1 |
| Ativação | sigmoidal | Tan hiperbólica | 0 |
| Atualização | Síncrona | Síncrona | 1 |
| Taxa de aprendizado | 0,4 | 0,5 | 0,81 |
| Momento | 0,1 | 0,2 | 0,61 |
| Total (dist. Euclidiana) | ---------- | ---------- | 0,848 |

**Tabela 7.1 - Comparação dos Casos**

| | Caso Recuperado | Caso Adaptado | Caso dos Alunos |
|---|---|---|---|
| Taxa de erro (MSE) | 13,9 % | 27,3% | 4,7% |

**Tabela 7.2 – Comparação dos desempenhos das RNAs**

## 7.3 Considerações Finais

A principal questão em experimentos envolvendo aplicações de RBC é a garantia da qualidade dos casos contidos na base. Neste experimento, como não poderia deixar de ser, este aspecto foi extremamente influente, mas não ao ponto de inviabilizar qualquer tipo de análise.

Efetivamente, reunir um conjunto expressivo e balanceado de casos em um determinado domínio é uma tarefa difícil, não só pela necessidade de um conhecimento de um

especialista no domínio relacionado para decidir quais casos devem ser levados em conta, mas também pela própria dificuldade na obtenção desses casos. A utilização dos casos do projeto StatLog pode ter minimizado a questão da pertinência ou não dos casos, mas proporcionou ainda uma base reduzida. Nem todos os detalhes de implementação de RNAs estão disponíveis nas publicações. É necessário um contato direto como os modeladores das redes para obtê-los, conforme ficou claro no contato direto com os alunos que desenvolveram as soluções dos exercícios. Seria interessante, portanto, aumentar a quantidade de casos na base a partir deste núcleo estabelecido.

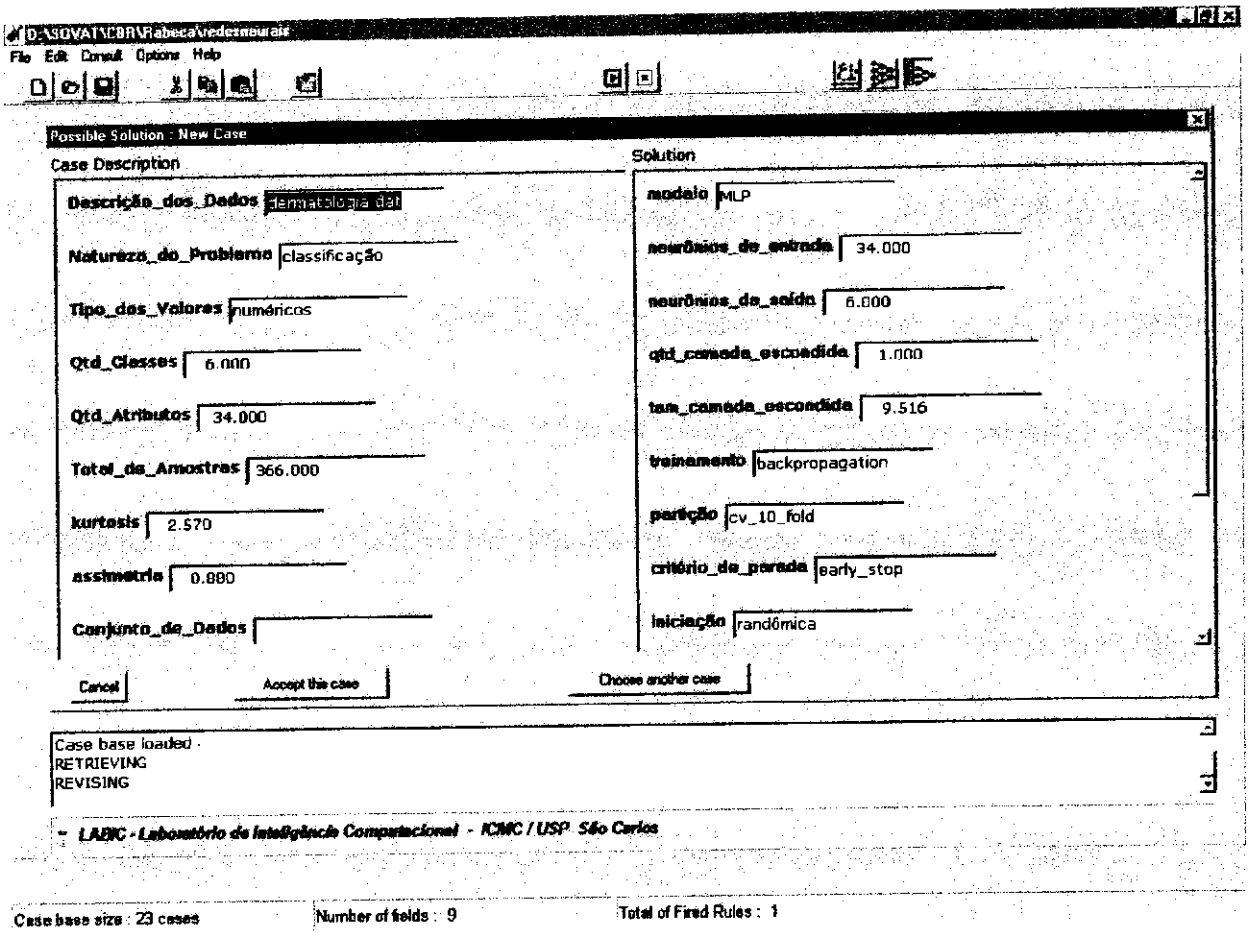


Figura 7.3 - Tela do Caso Novo (Adaptado)

Os conjuntos de regras de pré-processamento e de reparo, apesar de terem sido iniciados por pontos bastante influentes, podem ser enriquecidos com heurísticas e práticas usuais dos especialistas na área de RNA. Porém, no escopo deste trabalho, os conjuntos empregados podem ser considerados significativos. O mesmo pode ser dito a respeito das generalizações.

O estudo do caso adaptado pode ser melhor realizado se analisarmos a influência de cada atributo. Existe um grupo de atributos que influenciou pouco na qualidade da solução, por não apresentar diversidade na base e portanto não oferecer alternativas. A base terminou

por possuir 87% dos casos de MLP, ou seja, na verdade é uma base deste modelo. O exercício foi exatamente destinado a este mesmo modelo, logo a similaridade máxima obtida neste atributo era esperada. O mesmo aconteceu com o atributo que guarda o algoritmo de treinamento empregado, por estar diretamente associado ao modelo. Apesar de algumas soluções dos exercícios empregarem variações do *backpropagation*, o projeto StatLog escolheu especificamente esse algoritmo adicionado de termo de momento como objeto de seu estudo. Da mesma forma, o emprego do critério de parada por *early stop*, a iniciação randômica dos pesos e a atualização síncrona dos estados foram unânimes, em ambas as fontes de casos. Essas práticas porém, já se encontram muito solidificadas. À medida que variações não usuais de MLP forem adicionadas à base, estes atributos passarão a ter influência.

No que diz respeito ao estabelecimento da topologia, a similaridade máxima encontrada foi conseqüência direta da aplicação de uma regra que também foi praticada pelos alunos. Em alguns casos da base, porém, quando o número de classes é elevado, a regra que determina o tamanho da camada de saída não é aplicada. É mais eficiente representar a classe através de uma representação binária do número associado à classe, ou seja, a camada de saída possui uma quantidade de neurônios igual ao logaritmo do número de classes na base dois. Esses casos, porém eram poucos (3) na base, não tendo sido incluídos no topo da lista de casos recuperados. Contudo, o índice de similaridade indica uma boa escolha nessas regras de reparo. Além disso, ambas as fontes de casos trabalham com apenas uma camada escondida.

Por outro lado, os alunos não parecem ter aplicado a segunda regra de reparo, dando preferência a manter uma partição do tipo 50-25-25 para treinamento, validação e teste do que implementar uma técnica de *fold* em pequenos conjuntos de dados, como foi o caso do exercício. A similaridade nula neste caso indica pouca experiência por parte dos usuários, ainda que suas redes finais tenham tido um bom desempenho em seu melhor caso.

Outra similaridade nula ocorreu na escolha da função de ativação, já que as implementações da base, todas, empregam uma tangente hiperbólica como função, mais genérica para valores numéricos, enquanto a escolha nos exercícios recaiu sobre a tradicional função sigmoidal, que também era adequada ao problema em questão. É um valor que aponta para uma falta de experiência expressa na solução do exercício.

Finalmente, três parâmetros usualmente empíricos apresentaram as similaridades intermediárias.

O tamanho da camada escondida parece ter sido escolhido sem qualquer critério na solução do exercício pelos alunos, indicando apenas uma tendência em usar o máximo de neurônios enquanto o treinamento fosse convergente. O caso adaptado pelo sistema lançou

mão da regra de pré-processamento, escolhendo o tamanho da camada oculta baseado na quantidade de amostras. O resultado não inteiro (9,52) obtido pelo sistema deve-se ao fato de o RaBeCa ainda não possuir uma função que arredonde um número real para um inteiro. Os valores da taxa de aprendizado e termo de momento, apesar de aproximarem-se, por tratarem-se de valores usuais, proporcionaram um casamento apenas aproximado.

A discrepância no desempenho das redes pode ter sofrido a influência de incorreções no processo de simulação, que não foi totalmente descrito quanto a procedimentos de normalização e que foi iniciado randomicamente. Seria interessante realizar uma execução sistemática do simulador com pequenas variações em torno do ponto sugerido de modo a dimensionar a variação no desempenho da rede em função da mudança desses parâmetros. Essa e outras sugestões serão incluídas no capítulo a seguir.

# Capítulo 8

# Conclusão e Contribuições

Nesta tese, foram abordados alguns aspectos de RBC, um dos braços da área de Inteligência Artificial. Entre estes aspectos encontram-se sua metodologia bem definida, suas características adequadas à hibridização e integração com os demais tópicos da área e seu potencial para o tratamento do problema de escolha e configuração de modelos de RNAs, especificamente. Neste último caso, foi delineado um método para aplicar RBC ao problema. Além disso, foi proposta uma hibridização de modelos conexionistas com RBC em sua etapa de recuperação e adaptação. Para possibilitar a avaliação destas propostas e técnicas foi implementado um ambiente de desenvolvimento de RBC. Neste capítulo, serão comentadas as contribuições principais do trabalho, ressaltadas as suas limitações e indicados trabalhos futuros em continuação do tema.

## 8.1 Contribuições

Inicialmente foi realizada uma explanação das características que tornam a metodologia RBC adequada ao tratamento do problema de escolha e configuração de RNAs. Esta abordagem não foi tentada anteriormente, constituindo-se em uma abordagem atraente para a área de sistemas conexionistas, sobretudo no tocante a ensino de estudantes interessados por RNAs. Esta proposta de abordagem do domínio de RNAs por RBC foi submetida e aprovada para apresentação em *poster* durante o último Encontro Nacional de Inteligência Artificial, ENIA 2001 [Sovat, Carvalho 2001a].

No Capítulo 2 foi feita uma descrição do que se compreende por RBC e foram ressaltadas suas etapas principais. Dessa maneira, foi gerado um resumo bastante prático e atualizado dos conceitos que estruturam RBC, com uma ênfase na etapa de adaptação, que, juntamente com problemas de manutenção da base de casos, cada vez mais se torna o tema das pesquisas na área.

No Capítulo 3, além de uma breve revisão da taxonomia existente na literatura sobre sistemas híbridos reunindo RNAs e sistemas simbólicos (chamados aqui de neuro-simbólicos), foi proposto um mecansimo pelo qual um sistema classificado nesta taxonomia pode ser empregado para aumentar o poder de generalização das etapas de recuperação e, principalmente, de adaptação. Foi realizada uma breve caracterização de como o domínio proposto pode ser moldado para ser tratado por RBC. Levantamentos de aspectos relevantes no projeto de uma rede neural contribuem para uma abordagem mais formalizada da área, numa tentativa de reduzir seu aspecto empírico e, mais uma vez, facilitar a prática do ensino de disciplinas correlatas. Apesar de ter sido descrito em detalhes o processo de implementação daquele mecanismo, não foi possível incluir-se neste trabalho um experimento que pudesse testar seu desempenho, em virtude da grande carga necessária de desenvolvimento de *software*, escapando ao escopo de tempo da tese. Porém, as ferramentas básicas foram implementadas no interior do ambiente descrito no Capítulo 4 e a confecção deste experimento está prevista em trabalhos futuros.

Cabe ressaltar que esta proposta de um mecanismo híbrido na recuperação e adaptação de casos foi submetida para revisão e aceita para apresentação oral na 4ª Conferência Internacional de RBC [Sovat, Carvalho 2001b], especificamente no *workshop* destinado a técnicas de *soft computing*, como é denominada a fusão de técnicas de Inteligência Artifical com os algoritmos convencionais. Os primeiros resultados de experimentos envolvendo este tópico serão também submetidos para publicação em periódicos de *soft computing* (Computational Intelligence).

O Capítulo 4 apresenta uma nova ferramenta para desenvolvimento e experimentação em RBC. Esta ferramenta, RaBeCa, é um ambiente de desenvolvimento (*shell*) que roda em MS-Windows e implementa todas as fases de um ciclo RBC. Ela foi implementada totalmente durante o desenrolar deste trabalho e encontra-se disponível em meio magnético. Além da motivação de prover um meio de poder executar experimentos envolvendo diversas abordagens de Inteligência Artificial, esta iniciativa também se deveu ao fato de não existir programas análogos gratuitos disponíveis com facilidade para a comunidade acadêmica. Essa carência é assunto recorrente em listas de discussão e espera-se que venha a ser mitigada com mais esta contribuição.

Artigos descrevendo diversos aspectos e etapas de desenvolvimento do RaBeCa foram submetidos e aprovados para apresentação em eventos nacionais (ENIA 99 e 01) [Sovat, Carvalho 1999], [Sovat, Carvalho 2001a], e internacionais (ICCIMA 2001) [Sovat, Carvalho

2001c] e (ICTAI 2001) [Sovat et al. 2001]. O trabalho concluído também foi submetido para publicação na revista "International Journal on Artificial Intelligence Tools (IJAIT)".

O Capítulo 5 mostra como pode ser realizada a montagem de uma base de casos de modelos de RNAs. Foram descritos e justificados os atributos escolhidos, as regras e generalizações empregadas. As limitações encontradas serão comentadas na próxima seção e uma listagem da base de casos criada encontra-se no Apêndice A.

No Capítulo 6 é apresentada uma forma de avaliação de sistemas RBC, que sempre é um ponto de difícil abordagem na área. Uma breve proposta de metodologia para avaliação do sistema híbrido constitui um recurso a ser aproveitado em trabalhos futuros. As seções também fornecem informações sobre o que exatamente deve-se buscar medir quando examinando sistemas RBC.

O Capítulo 7 descreve um estudo de caso simples, porém eficaz, que mostra a aplicação de uma das idéias contidas na tese. Seus resultados são analisados e possíveis correlações entre a estrutura da base de casos e as escolhas dos usuários são levantadas.

Basicamente, este trabalho pretende estabelecer uma área de ensaios inicial para que aconteça uma maior integração entre sistemas conexionistas e simbólicos. Esta integração pode ser híbrida, como no mecanismo de memória associativa proposto, ou uma aplicação ao domínio, como no caso do experimento do Capítulo 7. De qualquer forma, contribui para um ponto de discussão, sempre necessário, envolvendo esses dois mundos.

## 8.2 Limitações

Sem dúvida, a maior limitação deste trabalho deveu-se à dificuldade em construir uma base de casos de grande porte. Contudo, este trabalho é essencialmente incremental e deve continuar sendo desenvolvido no ambiente do LABIC, com a introdução de novos modelos, escolhas de parâmetros e estratégias de treinamento, já que cada vez mais se tem acesso a especialistas da área, essenciais para esta tarefa.

O foco principal da tese também terminou por deslocar-se de sistemas conexionistas propriamente ditos para concentrar-se nas possibilidades do ciclo RBC. Um maior estudo do comportamento real de RNAs é necessário. Seguindo este objetivo, é desejável um estudo sistemático das conseqüências de pequenas variações nos parâmetros escolhidos para cada modelo e o desempenho final da RNA. Este estudo, agora factível através do esquema montado para o experimento do Capítulo 7, fugiu ao escopo deste trabalho, mas sem dúvida possui agora uma excelente oportunidade de ser implementado em um trabalho futuro.

Não foram também analisadas questões relativas a manutenção de bases de casos, já que tal análise, além de abrir toda uma nova vertente de pesquisas, exigiria mais uma vez a construção de uma base de casos de grande porte.

O experimento do Capítulo 7 foi bastante afetado pela carência de diversidades já apontada. Suas limitações envolveram também a quantidade de consultas à base de casos e a aplicação de outras métricas diferentes da métrica de eficiência.

É importante ressaltar, porém, que estas limitações, apesar de obviamente reduzirem o alcance do presente trabalho, não inviabilizam sua continuidade. Em vez disso, elas foram encaradas como um primeiro passo dentro de um objetivo de pesquisa contínuo.

## 8.3 Trabalhos Futuros

Além dos trabalhos futuros já referidos nas duas seções anteriores deste capítulo, são antevistos também alguns pontos desejáveis para a continuação da pesquisa.

No que diz respeito ao RaBeCa, além da hibridização estudada é importante enfatizar outros potenciais cruzamentos, como, por exemplo, uma possível utilização de lógica nebulosa (*fuzzy logic*) em critérios de similaridade e regras de adaptação; técnicas de linguagem natural na descrição dos casos e representações mais robustas de planos e *scripts* nas soluções e métodos indutivos na indexação dos casos.

Como pontos para implementação de novas facilidades no RaBeCa, a curto prazo, incluem-se a possibilidade de edição da base de casos dentro do ambiente, com facilidades para medição do grau de variedade dos casos; edição das regras de pré-processamento e adaptação; interfaceamento com outros formatos de arquivo para saída de soluções e finalmente, conforme as idéias expostas, opções de configuração de modelos de indexação, adaptação e descrição dos casos.

A fim de permitir uma maior facilidade para o usuário do RaBeCa poder testar alternativas diversas dentro do ambiente e incluir seus próprios algoritmos, seria conveniente aproveitar a estrutura de objetos do programa e adaptá-la para uma abordagem orientada a componentes, ou objetos COM, da mesma forma que já foram realizadas alterações no projeto do Kipu [Sousa 2000]. Desta maneira, por exemplo, usuários externos poderiam incluir diferentes critérios de similaridade ou novas classes de atributos conforme desejassem.

Além disso, algumas sugestões envolvendo o tratamento de RNAs por RBC envolvem:

abordar o problema de projetar uma rede neural como planejamento, descrevendo todo o experimento em etapas, lançando mão de outras representações (scripts, por exemplo).

análises com métricas diversas das de sistemas de informação, sobretudo uma análise da usabilidade da interface do usuário do sistema final

automação total da etapa de reutilização, com uma realimentação automática do simulador para o RaBeCa, de modo a aumentar o poder da análise sistemática imaginada anteriormente.

testar o sistema com alunos da disciplina Redes Neurais em um próximo período letivo no ICMC, de modo a medir o desempenho exatamente com o tipo de usuário que foi imaginado.

Finalmente, é necessário reafirmar o objetivo de continuar o estudo de sistemas híbridos, dentro de uma postura que os considera como uma abordagem promissora para a solução de problemas reais, contribuindo para a disseminação das técnicas de Inteligência Artificial cada vez mais no ambiente da computação.

# Apêndice A

# Listagem da Base de Casos

```
introduction is

      'Esta é uma base de casos contendo exemplares',
      'de modelos de redes neurais implementados  em',
      'tarefas relacionadas ao reconhecimento de padrões',
      'e a classificadores',
      ''

end;


case definition is

      field Descrição_dos_Dados type is string weight is 0;

      field Natureza_do_Problema type is
            (classificação, regressão, extração_de_características,
clusterização) weight is 2;

      field Tipo_dos_Valores type is
            (numéricos, simbólicos_não_ordenados, simbólicos_ordenados)
weight is 2;

      field Qtd_Classes type is number weight is 2;

      field Qtd_Atributos type is number;

      field Total_de_Amostras type is number;

      field kurtosis type is number;

      field assimetria type is number;

      field Conjunto_de_Dados type is string;

end;


index definition is

      index on Natureza_do_Problema;
      index on Tipo_dos_valores;

end;


modification definition is

      abstraction regressão is (previsão, ajuste);
      abstraction clusterização is (clustering, agrupamento);

      field Total_de_Amostras similar range 100 to 500;
      field Total_de_Amostras similar range 500 to 1000;
      field Total_de_Amostras similar range 1000 to 5000;
      field Total_de_Amostras similar range 5000 to 20000;
```

```
end;

preprocess rule definition is

repair rule testa_exec is

      when
            kurtosis is undefined
      then
            execute "CalcKurt.exe";
      end;

repair rule testa_exec is

      when
            assimetria is undefined
      then
            execute "CalcAssim.exe";
      end;

end;


repair rule definition is

      repair rule ajusta_topologia is
      when
            1 = 1 ~ sempre executa
      then
            evaluate neurônios_de_entrada to qtd_atributos;
            evaluate neurônios_de_saída to qtd_classes;
            evaluate tam_camada_escondida to ln(Total_de_amostras)/ln(2)+1;

      end;

      repair rule tipo_validacao is
      when
          Total_de_amostras < 1000
      then
          change partição to "cv_10_fold";
      end;
end;

~casos da base
case instance rede_1 is

Descrição_dos_dados = 'Análise química de vinhos';
Natureza_do_Problema = agrupamento;
tipo_dos_valores = numéricos ;
Total_de_amostras = 178;
Qtd_Atributos = 13;
Qtd_Classes = 3;
conjunto_de_dados = 'vinhos.dat';

solution is

modelo = SOM;
neurônios_de_entrada = 13;

alfa = 0.9;
raio = 0.001;
largura = 20;
treinamento = Kohonen;
critério_de_parada = total_de_ciclos;
iniciação = randômica;
```

```
end;

case instance rede_2 is

Descrição_dos_dados = 'Análise de crédito - CredMan';
Natureza_do_Problema = classificação;
tipo_dos_valores = enumerados_e_numéricos ;
Total_de_amostras = 20000;
Qtd_Atributos = 7;
Qtd_Classes = 2;
kurtosis = 93.1399;
assimetria = 6.1012;
conjunto_de_dados = 'german.dat';

solution is

modelo = MLP;
neurônios_de_entrada = 7;
neurônios_de_saída = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 2;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.5;
momento = 0.1;

end;

case instance rede_3 is

Descrição_dos_dados = 'Análise de crédito - CrAust';
Natureza_do_Problema = classificação;
Qtd_Atributos = 14;
Qtd_Classes = 2;
tipo_dos_valores = enumerados_e_numéricos ;
Total_de_amostras = 690;
kurtosis = 12.5538;
assimetria = 1.9701;

solution is

modelo = MLP;
neurônios_de_entrada = 14;
neurônios_de_saída = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 3;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.4;
momento = 0.1;

end;

case instance rede_4 is

Descrição_dos_dados = 'Algarismos Manuscritos - Dig44';
Natureza_do_Problema = classificação;
Qtd_Atributos = 16;
```

```
Qtd_Classes = 10;
tipo_dos_valores = numéricos;
Total_de_amostras = 18000;
kurtosis = 5.1256;
assimetria = 0.8562;

solution is

modelo = MLP;
neurônios_de_entrada = 16;
neurônios_de_saída = 10;
qtd_camada_escondida = 1;
tam_camada_escondida = 4;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.5;
momento = 0.1;

end;

case instance rede_5 is

Descrição_dos_dados = 'Algarismos Karhunen-Loeve - KL';
Natureza_do_Problema = classificação;
Qtd_Atributos = 40;
Qtd_Classes = 10;
tipo_dos_valores = numéricos;
Total_de_amostras = 18000;
kurtosis = 2.92;
assimetria = 0.1802;

solution is

modelo = MLP;
neurônios_de_entrada = 40;
neurônios_de_saída = 10;
qtd_camada_escondida = 1;
tam_camada_escondida = 5;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.5;

end;

case instance rede_6 is

Descrição_dos_dados = 'Silhuetas de Veículos';
Natureza_do_Problema = classificação;
Qtd_Atributos = 18;
Qtd_Classes = 4;
tipo_dos_valores = numéricos ;
Total_de_amostras = 846;
kurtosis = 5.18;
assimetria = 0.8282;

solution is

modelo = MLP;
neurônios_de_entrada = 18;
```

```
neurônios_de_saída = 4;
qtd_camada_escondida = 1;
tam_camada_escondida = 5;
treinamento = backpropagation;
partição = 'cv_10_fold';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.5;
momento = 0.2;

end;

case instance rede_7 is

Descrição_dos_dados = 'Reconhecimento de letras';
Natureza_do_Problema = classificação;
Qtd_Atributos = 16;
Qtd_Classes = 26;
tipo_dos_valores = numéricos ;
Total_de_amostras = 20000;
kurtosis = 3.5385;
assimetria = 0.5698;

solution is

modelo = MLP;
neurônios_de_entrada = 16;
neurônios_de_saída = 5;
qtd_camada_escondida = 1;
tam_camada_escondida = 5;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.5;
momento = 0.09;

end;

case instance rede_8 is

Descrição_dos_dados = 'Cromossomos';
Natureza_do_Problema = classificação;
Qtd_Atributos = 16;
Qtd_Classes = 24;
tipo_dos_valores = numéricos ;
Total_de_amostras = 40000;
kurtosis = 4.4024;
assimetria = 0.42;

solution is

modelo = RBF;

end;

case instance rede_9 is

Descrição_dos_dados = 'Imagens do LandSat';
Natureza_do_Problema = classificação;
Qtd_Atributos = 36;
Qtd_Classes = 6;
Total_de_amostras = 6435;
```

```
tipo_dos_valores = numéricos ;
kurtosis = 4.1737;
assimetria = 0.7316;

solution is

modelo = MLP;
neurônios_de_entrada = 7;
neurônios_de_saída = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 3;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.4;
momento = 0.1;

end;

case instance rede_10 is

Descrição_dos_dados = 'Segmentação de Imagens';
Natureza_do_Problema = classificação;
Qtd_Atributos = 11;
Qtd_Classes = 7;
Total_de_amostras = 2310;
tipo_dos_valores = numéricos ;
kurtosis = 24.4813;
assimetria = 2.9580;

solution is

modelo = MLP;
neurônios_de_entrada = 11;
neurônios_de_saída = 7;
qtd_camada_escondida = 1;
tam_camada_escondida = 3;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.6;
momento = 0.2;

end;

case instance rede_11 is

Descrição_dos_dados = 'Corte20';
Natureza_do_Problema = classificação;
Qtd_Atributos = 20;
Qtd_Classes = 2;
Total_de_amostras = 18700;
tipo_dos_valores = numéricos ;
kurtosis = 3.5214;
assimetria = 0.9012;

solution is

modelo = MLP;
neurônios_de_entrada = 20;
neurônios_de_saída = 2;
```

```
qtd_camada_escondida = 1;
tam_camada_escondida = 2;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.7;
momento = 0.2;

end;

case instance rede_12 is

Descrição_dos_dados = 'Corte50';
Natureza_do_Problema = classificação;
Qtd_Atributos = 50;
Total_de_amostras = 18700;
tipo_dos_valores = numéricos ;
Qtd_Classes = 2;

solution is

modelo = MLP;
neurônios_de_entrada = 50;
neurônios_de_saida = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 2;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.7;
momento = 0.2;

end;

case instance rede_13 is

Descrição_dos_dados = 'Ferimentos na Cabeça';
Natureza_do_Problema = classificação;
Qtd_Atributos = 6;
Qtd_Classes = 3;
Total_de_amostras = 900;
tipo_dos_valores = numéricos;
kurtosis = 5.0408;
assimetria = 1.0071;

solution is

modelo = MLP;
neurônios_de_entrada = 6;
neurônios_de_saída = 3;
qtd_camada_escondida = 1;
tam_camada_escondida = 10;
treinamento = backpropagation;
partição = 'cv_9_fold';
critério_de_parada = early_stop;

iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.4;
momento = 0.1;
```

```
end;

case instance rede_14 is

Descrição_dos_dados = 'Doenças cardíacas';
Natureza_do_Problema = classificação;
Qtd_Atributos = 13;
Qtd_Classes = 2;
tipo_dos_valores = numéricos ;
Total_de_amostras = 270;
kurtosis = 3.6494;
assimetria = 0.956;

solution is

modelo = MLP;
neurônios_de_entrada = 13;
neurônios_de_saída = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 7;
treinamento = backpropagation;
partição = 'cv_9_fold';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.4;
momento = 0.05;

end;

case instance rede_15 is

Descrição_dos_dados = 'Análise de Crédito - CrGer';
Natureza_do_Problema = classificação;
Qtd_Atributos = 24;
Qtd_Classes = 2;
Total_de_amostras = 1000;
tipo_dos_valores = numéricos ;
kurtosis = 7.7943;
assimetria = 1.6986;

solution is

modelo = MLP;
neurônios_de_entrada = 24;
neurônios_de_saída = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 11;
treinamento = backpropagation;
partição = 'cv_10_fold';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.6;

end;


case instance rede_16 is

Descrição_dos_dados = 'Controle do Shuttle';
Natureza_do_Problema = classificação;
Qtd_Atributos = 9;
Qtd_Classes = 7;
```

```
Total_de_amostras = 58000;
tipo_dos_valores = numéricos ;
kurtosis = 160.3108;
assimetria = 4.4371;

solution is

modelo = MLP;
neurônios_de_entrada = 9;
neurônios_de_saída = 7;
qtd_camada_escondida = 1;
tam_camada_escondida = 4;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.4;
momento = 0.05;

end;

case instance rede_17 is

Descrição_dos_dados = 'Diabetes';
Natureza_do_Problema = classificação;
Qtd_Atributos = 8;
Qtd_Classes = 2;
Total_de_amostras = 768;
tipo_dos_valores = numéricos ;
kurtosis = 5.8270;
assimetria = 1.0586;

solution is

modelo = MLP;
neurônios_de_entrada = 8;
neurônios_de_saída = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 10;
treinamento = backpropagation;
partição = 'cv_12_fold';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;

end;

case instance rede_18 is

Descrição_dos_dados = 'DNA';
Natureza_do_Problema = classificação;
Qtd_Atributos = 60;
Qtd_Classes = 3;
tipo_dos_valores = numéricos_e_simbólicos ;
kurtosis = 29.5674;
assimetria = 2.5582;


solution is

modelo = RBF;
centros = 720;

end;
```

```
case instance rede_19 is

Descrição_dos_dados = 'Técnico';
Natureza_do_Problema = classificação;
Qtd_Atributos = 56;
Qtd_Classes = 91;
Total_de_amostras = 7080;
tipo_dos_valores = numéricos;
kurtosis = 108.2963;
assimetria = 6.7156;

solution is

modelo = RBF;

end;

case instance rede_20 is

Descrição_dos_dados = 'Sistema de potência Belga 1';
Natureza_do_Problema = classificação;
Qtd_Atributos = 28;
Qtd_Classes = 2;
Total_de_amostras = 2500;
tipo_dos_valores = numéricos;
kurtosis = 2.6581;
assimetria = 0.4334;

solution is

modelo = MLP;
neurônios_de_entrada = 28;
neurônios_de_saída = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 12;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;

end;

case instance rede_21 is

Descrição_dos_dados = 'Sistema de potência Belga 2';
Natureza_do_Problema = classificação;
Qtd_Atributos = 57;
Qtd_Classes = 2;
Total_de_amostras = 3000;
tipo_dos_valores = numéricos;
kurtosis = 6.7738;
assimetria = 1.1180;

solution is

modelo = MLP;
neurônios_de_entrada = 7;
neurônios_de_saída = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 12;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
```

```
ativação = sigmóide;
atualização = síncrona;
end;

case instance rede_22 is

Descrição_dos_dados = 'Falhas em Máquinas';
Natureza_do_Problema = classificação;
Qtd_Atributos = 45;
Qtd_Classes = 3;
Total_de_amostras = 570;
tipo_dos_valores = numéricos;
kurtosis = 6.9866;
assimetria = 1.8972;

solution is

modelo = MLP;
neurônios_de_entrada = 45;
neurônios_de_saída = 3;
qtd_camada_escondida = 1;
tam_camada_escondida   8;
treinamento = backpropagation;
partição = 'cv_10_fold';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.5;
momento = 0.1;
end;

case instance rede_23 is

Descrição_dos_dados = 'Distribuição da Mosca Tsé-Tsé';
Natureza_do_Problema = classificação;
Qtd Atributos = 14;
Qtd_Classes = 2;
Total_de_amostras = 4599;
tipo_dos_valores = numéricos;
kurtosis = 4.3322;
assimetria = 0.6483;

solution is

modelo = MLP;
neurônios_de_entrada = 14;
neurônios_de_saída = 2;
qtd_camada_escondida = 1;
tam_camada_escondida = 17;
treinamento = backpropagation;
partição = '75-25-25';
critério_de_parada = early_stop;
iniciação = randômica;
ativação = sigmóide;
atualização = síncrona;
taxa_aprendizado = 0.5;

end;
```

# Apêndice B

# Sintaxe da Linguagem CASL

Esta é a sintaxe original da linguagem CASL acrescida de algumas extensões criadas ao longo desta tese.

Original: `CASL description document v2 13 © UW Aberystwyth, September 1996;`

Estendida: `Dezembro de 2001.`

## B.1 Módulos da Base de Casos

Um script em CASL representa uma base de casos quando expressa uma seqüência *caseseq*, descrita a seguir:

*caseseq ::= introblock defns indexblock* $\{modblock\}_{opt}$ $\{preprocessblock\}_{opt}$ $\{repblock\}_{opt}$ *cases*;

## B.2 Identificação da Base

*introblock ::=* **introduction is** *introseq* **end ;**
|
;
*introseq ::= string*
| *string , introseq*
;
*string ::= "seq_char"*

*seq_char ::=* $\{char\}_{0-}$

*char ::=* qualquer letra, mesmo acentuada, ou algarismo.

## B.3 Definição dos atributos de caso

*defns ::=* **case definition is** *fielddefs* **end ;**
;
*fielddefs ::= fielddef fielddefs*
| *fielddef*
;
*fielddef ::=* **field** *identifier* **type is** *string weightdef promptdef* ;
| **field** *identifier* **type is** *number weightdef promptdef* ;
| **field** *identifier* **type is** *enumeration weightdef promptdef orddef* ;

| **field** *identifier* **type is** *list weightdef promptdef* **;**
;
*weightdef ::=* **weight is** *realnum*
|
;
*promptdef ::=* **prompt is** *listbox*
|
;
*orddef ::=* **ordered**
|
;
*enumeration ::=* **(** *identlist* **)**
;
*identlist ::= identifier*
| *identifier , identlist*
;

B.4 Atributos índices

*indexblock ::=* **index definition is** *indexlist* **end ;**
;
*indexlist ::= index*
| *index indexlist*
;
*index ::=* **index on** *identifier* **;**
;

B.5 Generalizações (abstrações e intervalos de similaridade)

*modblock ::=* **modification definition is** *modrules* **end ;**
|
;
*modrules ::= modrule modrules*
| *modrule*
;

*modrule ::=* **abstraction identifier is** *enumeration* **;**
| **field** *identifier* **similar range** *realnum* **to** *realnum* **;**
| **field** *identifier* **similar range -** *realnum* **to** *realnum* **;**
| **field** *identifier* **similar range -** *realnum* **to -** *realnum* **;**
;

B.6 Regras

*preprocessblock ::=* **preprocess rule definition is** *repseq* **end ;**
|
;
*repblock ::=* **repair rule definition is** *repseq* **end ;**
|

```
;
repseq ::= reprule
        | reprule repseq
;

reprule ::= repair rule identifier is when combtree then replist end ;
;
combtree ::= combtree or combtree
        | combtree and combtree
        | ( combtree )
        | not combtree
        | combtreeitem
;
combtreeitem ::= identifier is identifier
        | expression compop expression
        | identifier is string
        | identifier is listbox
        | listop is identifier
        | listop compop expression
        | listop is string
        | listop is listbox
        | identifier is undefined
;
compop ::= =
        | !=
        | <
        | >
        | <=
        | >=
;
replist ::= replistitem
        | replistitem replist
;

replistitem ::= change identifier to identifier ;
        | evaluate identifier to expression ;
        | change identifier to string ;
        | change identifier to listbox ;
        | change listop to identifier ;
        | evaluate listop to expression ;
        | change listop to string ;
        | change listop to listbox ;
        | change identifier to undefined ;
        | change identifier extract identifier from identifier ;
        | repair ;
        | reselect ;
        | pr ( listbox ) ;
        | change weight of identifier to realnum ;
        | execute ( string );
 ;
```

*listop* ::= **hd** ( *listop2* )
| **tl** ( *listop2* )
;
*listop2* ::= *listop*
| *identifier*
;
*expression* ::= *expression* + *expression*
| *expression* - *expression*
| *expression* * *expression*
| *expression* / *expression*
| *expression* ^ *expression*
| ( *expression* )
| - *expression*
| *realnum*
| *identifier*
;

B.8 Definição dos casos

*cases* ::= *case*
| *case cases*
;
*case* ::= **case instance** *identifier* **is** *fieldlist local_fielddefs* **solution is** *fieldlist local_repairs* **end ;**
;
*local_fielddefs* ::= **local field definition is** *fielddefs*
|
;
*fieldlist* ::= *field*
| *field fieldlist*
;
*field* ::= *identifier* = *identifier* ;
| *identifier* = *realnum* ;
| *identifier* = - *realnum* ;
| *identifier* = *string* ;
| *identifier* = *listbox* ;
;
*listbox* ::= [ *list* ]
| [ ]
;

*list* ::= *listitem*
| *listitem list*
;
*listitem* ::= *identifier*
| *realnum*
| - *realnum*
| *string*
| *listbox*

;

*local_repairs* ::= **local repair rule definition is** *repseq*

|

;

# Apêndice C

# Descrição Funcional do RaBeCa

A seguir será realizada uma breve descrição funcional dos principais módulos de *software* componentes do ambiente de desenvolvimento. O RaBeCa foi implementado na linguagem Object Pascal, dentro do ambiente Delphi 4.0, segundo uma abordagem orientada a objetos. A Figura C.1 mostra a interligação lógica dos módulos, respeitando o fluxo de dados. O conteúdo de cada fluxo, bem como o que é entrada ou saída de cada módulo está relacionado juntamente com um resumo do algoritmo básico correspondente. Finalmente, serão listados os principais objetos e classes criados.

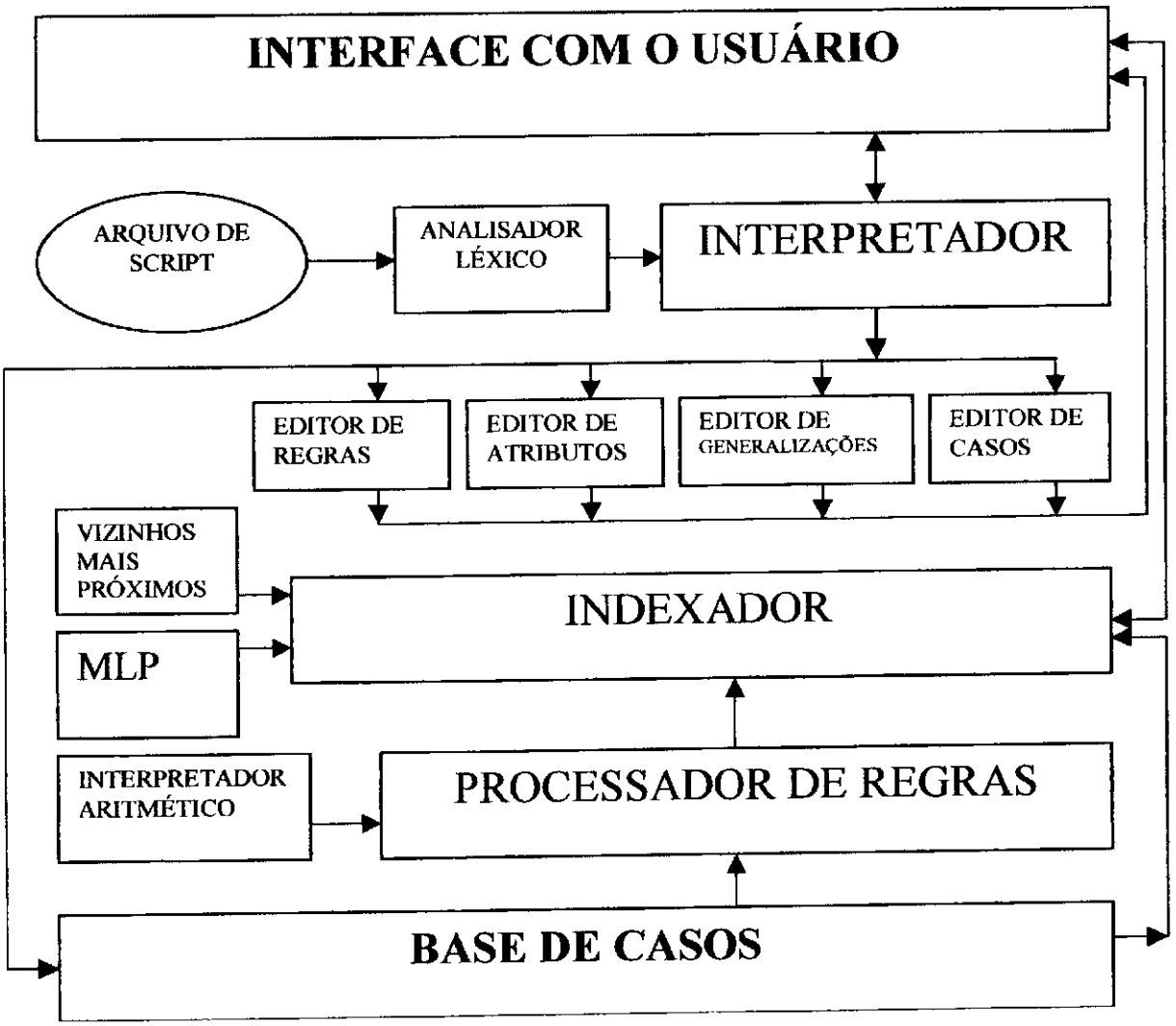


Figura C.1 - Diagrama de Fluxo dos Principais Módulos de *Software*

## C.1 Interpretador

O módulo Interpretador reúne as rotinas que implementam o analisador semântico do RaBeCa. Ele é responsável por construir as estruturas em memória correspondentes ao resultado da análise léxica do *script* em CASL, realizada pelos métodos implementados em um objeto da classe TScanner.

Este objeto é uma das duas únicas peças de *software* que não foram diretamente desenvolvidas para este trabalho. Seu principal método está descrito a seguir.

**procedure Analyze( Source: TStream )**

*Copyright (c) 1998 by Frank Plagge, Elsterweg 39, 38446 Wolfsburg, Germany.*

**Entrada**: uma estrutura *stream* (no caso, um arquivo texto, convertido para um objeto da classe TStream padrão do Object Pascal) .

**Saída**: nenhuma.

**Função**: instanciar um objeto da classe TToken, contendo uma seqüência de termos válidos encontrados (*tokens*). O total de tokens é foirnecido na propriedade *count*. Os termos em si são armazenados no vetor *Token*. Este vetor é destruído antes de cada nova análise.

A interface do Interpretador com o Analisador Léxico se dá através da chamada da rotina Scan:

**procedure Scan (var Nome : string);**

**Entrada:** o nome do arquivo contendo a base de casos;

**Saída**: nenhuma;

**Função:** instanciar os diversos objetos que representam a base de casos;

As demais rotinas deste módulo possuem cada uma o nome de cada seção possível em CASL, conforme visto no apêndice anterior. Além destas rotinas, foram implementados tratamentos de exceções para indicação de erros e de avisos.

Este módulo fornece os objetos necessários para que os quatro editores do RaBeCa (Atributos, Regras, Generalizações e Casos) apresentem suas respectivas informações ao usuário. Além disso, estes mesmos objetos compõem a Base de Casos na memória.

## C.2 Indexador

O módulo Indexador compreende rotinas que se relacionam com quatro atividades: exibição de campos do caso (atributos de caso); exibição de casos (recuperados ou adaptados); montagem e treinamento da memória associativa; consulta à base de casos.

Suas principais rotinas são:

**procedure MostraCampos;**

**Entrada**: nenhuma;

**Saída**: nenhuma;

**Função**: percorrer a instância da classe TConjuntoDeCampos e construir na tela uma janela contendo os componentes necessários para que o usuário forneça os valores de seu caso.

**procedure MostraCaso (Caso : TCaso);**

**Entrada**: um objeto da classe TCaso;

**Saída**: nenhuma;

**Função**: percorrer a instância da classe TCaso e construir na tela uma janela contendo os componentes necessários para a exibição dos atributos do caso.

**procedure MontaMemoriaAssociativa;**

**Entrada**: nenhuma;

**Saída**: nenhuma;

**Função**: instanciar um objeto da classe TMLP, segundo os seguintes passos:

1. Escolhe o número de camadas (3);
2. Faz tamanho da camada de entrada = quantidade de campos do caso;
3. Faz tamanho da camada de saída = quantidade de soluções conhecidas;
4. Faz tamanho da camada de entrada = média ponderada das anteriores;
5. Escolhe a função de ativação (tangente hiperbólica)
6. Escolhe a função de iniciação dos neurônios (randômica)
7. Escolhe a taxa de aprendizado (0,5);
8. Escolhe o termo de momento (0,9);
9. Escolhe o critério de parada (early stop);
10. Escolhe o período de validação (10 exemplares);
11. Escolhe tipo de treinamento (batch).

Esta função é chamada pelo módulo interpretador logo após a carga da base, caso a opção pelo uso da memória esteja ativada.

A consulta à base de casos foi implementada em um evento associado a um botão na tela. Este evento executa os seguintes passos:

1. Instancia um objetoda classe Tcomparador;
2. Executa o método Executa do objeto RegrasPrcPro( classe TconjuntoDeRegras) sobre o caso do usuário;
3. Se a memória associativa não está ativada
4. então chama a rotina VizinhosMaisProximos;
5. senão executa o método Executa do objeto MemAssoc (classe TMLP);
6. Se o número de vizinhos for diferente de zero
7. então faz
8. se existirem campos locais no primeiro vizinho (caso) da lista, constrói uma janela para consulta ao usuário sobre estes campos;
9. chama a rotina MostraCaso;
10. senão avisa ao usuário que não existem casos satisfatórios.

Neste módulo estão, portanto, as funções que criam a lista de casos recuperados, de acordo com o método escolhido (k-NN ou MLP). O funcionamento da MLP está descrito no Capítulo 3. A rotina VizinhosMaisProximos referida acima, executa, simplificadamente, os seguintes passos:

**procedure VizinhosMaisProximos;**

**Entradas**: quantidade de vizinhos desejada;
objeto da classe TCaso contendo o caso do usuário;

**Saída**: vetor de objetos da classe TCaso, contendo o vizinhos;

**Função**: basicamente segue o seguinte algoritmo:

1. Percorre os casos (vetor Instancias) fazendo
2. Se o caso possui um valor para os atributos índice iguais aos do caso do usuário
3. então seleciona o caso;
4. senão vai para o próximo caso;
5. Se selecionou o caso
6. Então chama a rotina Match, que calcula a similaridade do caso baseada nos métodos da instância da classe TComparador iniciada anteriormente;
7. Coloca o caso na lista de casos recuperados;
8. Ordena a lista pela similaridade.

### C.3 Processador de Regras

O módulo Regras engloba as rotinas que implementam os métodos da classe TConjuntoDeRegras. Existem objetos desta classe no Rabeca, correspondendo às regras de pré-processamento, às regras de reparo e a cada regra local existente. As rotinas implementam os sete tipos de ações possíveis no conseqüente de uma regra, segundo a sintaxe:

- Mudança de valor de um atributo (CHANGE)
- Extração de um valor de atributo de uma lista (EXTRACT)
- Avaliação de uma expressão aritmética (EVALUATE)
  - Estas três primeiras rotinas alteram os atributos de uma variável global CasoCorrente que aponta para umobjeto da classe TCaso. Para o EVALUATE, a rotina recorre ao Interpretador Aritmético. Este interpretador é a segunda e última peça de *software* que não foi desenvolveida para este trabalho. É implementado por um componente da classe TRapidEvaluator.
- Execução de programa (EXECUTE)
  - Esta rotina recorre ao sistema operacional para executar um programa externo.
- Disparo das regras de reparo globais (REPAIR)
  - Executa novamente o método Executa da instância da classe TConjuntoDeRegras associada às regras de reparo.
- Abandono de caso (RESELECT)
  - Interrompe a exibição do caso e passa para o próximo da lista.
- Exibição de mensagem (PR)
  - Constrói uma janela exibindo a mensagem associada.

Todas estas rotinas recebem como entrada um ponteiro para uma estrutura em memória. Esta estrutura deve ser interpretada de acordo com a ação correspondente. Não há nenhuma variável de saída.

Além disso, o módulo possui as rotinas de tratamento de extração de cabeça e cauda de uma lista (HD e TL).

### C.4 Principais Classes e Objetos

Estes são as principais classes citadas anteriormente, descritas na linguagem de implementação, Object Pascal.

```
TValor = class (TObject)
   Tipo : TipoAtributo;
   Item : pointer;
   constructor Create;
   destructor Destroy;
public
Tag  : longint;
   end;

TAtributo = class (TObject)
   Valor  : TValor;
   Caso   : integer;
   procedure Mostra (var NovoForm: TForm; var Pai : TWinControl; var LargLabel,
Alt : integer);
public
   Nome   : Str40;
   Peso   : double;
   Indice :  integer;
   Rotina :  TNotifyEvent;
end;

TConjuntoDeAtributos = class (TObject)
   QtdAtributos : integer;
   Nomes        : array of TAtributo;
   VMax, VMin   : array of extended;
   procedure Inclui (var Atributo : TAtributo);  overload;
   procedure Inclui (var Atributo : TAtributo; Posicao : integer); overload;
   procedure Remove (Num : integer);
public
end;
```

```
TCampo = class (TObject)
   Nome   : Str40;
   Tipo   : TStringList;
   Peso   : double;
   Prompt : TStringList;
   Caso   : integer;
private
public
end;

TConjuntoDeCampos = class (TObject)
   QtdCampos  : integer;
   Campos     : array of TCampo;
   procedure Inclui (var Campo : TCampo);
   function Indice (var Nome : Str40) : integer;
private
public
end;

TCaso = class (TObject)
   QtdAnonimos  : integer;
   Atributos    : TConjuntoDeAtributos;
   Solucao      : TConjuntoDeAtributos;
   CamposLocais : TConjuntoDeCampos;
   RegrasLocais : TConjuntoDeRegras;
   DoUsuario    : boolean;
   Similaridade : double;
   Nome         : Str40;
   Numero       : integer;
   constructor Create;
   destructor Destroy;
   function Achou (var Atributo : Str40) : TAtributo;
   procedure Mostra (var Janela : TForm);
 private
```

```
public
end;

TConjuntoDeCasos = class (TObject)
   QtdCasos  : integer;
   Casos     : array of TCaso;
   procedure Inclui (var Caso : TCaso);
   function Achou (var Nome : string) : TCaso;
private
public
end;

TRegra = class (TObject)
   Nome    : Str40;
   NumCaso : integer;
   Ativ    : boolean;
   Valor   : boolean;
   Ante    : PtrCombTree;
   Cons    : PtrAcao;
   constructor Create;
   destructor Destroy;
   procedure Executa (var Caso : TCaso);
private
public
end;

TConjuntoDeRegras = class (TObject)
   QtdRegras  : integer;
   Regras     : array of TRegra;
   procedure Executa (var Caso : TCaso);
   procedure Inclui (var Regra : TRegra);
private
public
end;
```

```
TEnumerados = class (TAtributo)
  constructor Create;
  destructor Destroy;
  procedure Mostra (var NovoForm: TForm; var Pai : TWinControl; var LargLabel,
Alt : integer);
private
LargMax : integer;
public
   Ordenado : boolean;
   Itens    : TStringList;
   Tag      : longint;
  end;

TNumericos = class (TAtributo)
  constructor Create;
  destructor Destroy;
  procedure Mostra (var NovoForm: TForm; var Pai : TWinControl; var LargLabel,
Alt : integer);
private
public
   Tag      : longint;
   Formato  : (inteiro, real_);
  end;

TStrings_ = class (TAtributo)
  constructor Create;
  destructor Destroy;
  procedure Mostra (var NovoForm: TForm; var Pai : TWinControl; var LargLabel,
Alt : integer);
private
public
  Tag      : longint;
  end;
```

```
TBooleanos = class (TAtributo)
  procedure Mostra (var NovoForm: TForm; var Pai : TWinControl; var LargLabel,
Alt : integer);
constructor Create;
  destructor Destroy;
private
public
  Tag     : longint;
  end;

TListas = class (TAtributo)
  constructor Create;
  destructor Destroy;
  procedure Mostra (var NovoForm: TForm; var Pai : TWinControl; var LargLabel,
Alt : integer);
private
LargMax : integer;
  public
  Tag    : longint;
  end;

TIntervalo = class (TObject)
private
public
   Ordem  : integer;
   Min    : integer;
   Max    : integer;
  end;

TAbstracao = class (TObject)
  procedure EscolheNovoValor;
private
public
```

```
    Nome    : Str40;
    Ultimo  : integer;
    Lista   : TStringList;
    Especie : TStringList;
  end;

TRange = class (TObject)
private
public
   Nome     : Str40;
    Range    : TList;
    MaxTotal,
    MinTotal  : integer;
  end;

TModificacoes = class (TObject)
  QtdAbstracoes : integer;
  QtdIntervalos : integer;
  Abstracoes    : array of TAbstracao;
  Intervalos    : array of TRange;
  procedure IncluiAbstracao (var Abstracao : TAbstracao);
  procedure IncluiIntervalo (var Intervalo : TRange);
  function AchouAbstracao (var Nome : String) : TAbstracao;
  function AchouNasAbstracoes (Valor : String; Lista : TStringList) : integer;
  function AchouIntervalo (var Nome : Str40) : TRange;
private
public
end;

TComparador = class (TObject)
private
public
  function Compara (var Atr1, Atr2 : TAtributo) : double; overload;
  function Compara (var Atr1, Atr2 : TEnumerados) : double; overload;
```

```
    function Compara (var Atr1, Atr2 : TNumericos) : double; overload;
    function Compara (var Atr1, Atr2 : TListas) : double; overload;
    function Compara (var Atr1, Atr2 : TStrings_) : double; overload;
    function Compara (var Atr1, Atr2 : TBooleanos) : double; overload;
end;

TMLP = class (TObject)
    QtdCamadas : integer;
    Camadas    : array of TCamada;
    SinaTemp   : MatrizDePesos;
    Sinapses   : MatrizDePesos;
    FAtivacao  : TAtivacao;
    constructor Create (NumCam : integer; var NumNeu : array of integer);
    procedure IncluiCamada (var Camada : TCamada);
    procedure IncluiSinapse (k, i, j : integer);
    function SinapseInicial (k : integer) : extended;
    function NET (k, j : integer) : extended;
    function Executa (var Entrada : Padrao) : Padrao;
    procedure Treina;
    procedure TreinaPadrao (var Entrada, Desejado: Padrao);
    function CalculaErro (Tipo : TipoErro): extended;
private
    TamBatch        : integer;
    ErroMinimo      : extended;
public
    TipoInicSinap   : TipoInicia;
    CriterioDeParada : TipoCriterio;
    TaxaDeAprendizado : extended;
    Momento         : extended;
    PeriodoValidacao : integer;
    TipoTreinamento  : TipoTreina;
end;
```

CasoDoUsuario: TCaso;

Informações do caso do usário

CasosRecuperados : ListadeCasos;

Lista de casos recuperados pelo indexador;

CasoReparado: TCaso;

O caso recuperado após a adaptação;

Instancias: TConjuntoDeCasos;

Lista de casos da base;

CamposDoCaso: TConjuntoDeCampos

Os atributos de caso previstos na base;

RegrasPrePro: TConjuntodeRegras;

Regras de pré-processamento;

RegrasReparo: TConjuntoDeRegras;

Regras de reparo (adaptação) globais;

Modificacoes: TModificacoes;

Generalizações;

# Referências Bibliográficas


Aamodt, A., Plaza, E., Case-based reasoning: Foundational Issues, Methodological Variations and System Approaches. *AI Communications*, 7(1) , 39—59, (1994).

Acorn, T. L., Walden S. H., SMART: Support management automated reasoning technology for COMPAQ customer service. In *Proceedings of the Fourth Annual Conference on Innovative Applications of Artificial Intelligence*, San Jose, CA, AAAI Press, 3—18, (1992).

Aha, D., Kibler D., Albert, M.K., Instance-Based Learning. *Machine Learning* 6(1), 37—66, (1991).

Althoff, K.-D, Wess, S., Case-based Reasoning and Expert System Development. In *Contemporary Knowledge Engineering and Cognition*, Springer-Verlag, 146—158, (1992).

Bareiss, R., Exemplar-based knowledge acquisition: a unified approach to concept representation, classification and learning. *Academic Press,*(1989).

Becker, L., Jazayeri, K., A Connectionist Approach to Case-based Reasoning. *DARPA Workshop on Case-Based Reasoning*, Hammond, K. J. Pensacola Beach, USA, 213—217. Morgan Kaufmann, (1989).

Carbonnel, J. G., Knoblock, C. A., Minton, S., Prodigy : an integrated architecture for planning and learning. *Architectures for Intelligence*, K. VanLehn, editor, Lawrence Erlbaum Associates, publishers, 241—278, (1991).

Carpenter, G., Grossberg, S., ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26, 4919—4930, (1987).

Casuel, CASUEL : A Common Case Representation Language. *ESPRIT Project Deliverable D1*, University of Kaiserslautern, Kaiserslautern, (1994).

Corchado, J. M., Lees, B. Adaptation of cases for case based forecasting with neural network support. In *Soft Computing in Case Based Reasoning*, chapter 13. Pal, S. K., Dillon, T. S., Yeung, D. S., eds. London, England, Springer-Verlag, (2000).

Dasarathy, B., Nearest neighbor norms: NN pattern classification techniques. *IEEE Computer Society Press,* (1990).

Domeshek, E. A case study of case indexing: designing index feature sets to suit task demands and support parallelism. *Advances in Connectionist and Neural Computation Theory, 2: Analogical Connections*, Bamden, J. and Holyoak, K.. Norwood, USA, (1993).

Ellis, D., *Modeling your own Neural Net*, IEEE Spectrum, Volume 34, Number 12, (1997).

ESTEEM, ESTEEM Enabling solutions through experience modelling. http://www.shai.com/PDF/ESTEEM.PDF, (2001).

Fabiunke, M., Kock, G., Milaré, C., Carvalho, A.C.P.L.F., Um Sistema Híbrido Integrando Características de Redes Neurais e Raciocínio Baseado em Casos. *Relatório Técnico*, ISSN-0103-2569, ICMC-USP. São Carlos (1997).

Gierl, L., Stengel-Rutkowski, S., Integrating consultation and semi-automatic knowledge acquisition in a prototype-based architecture: experiences with dysmorphic syndromes. *Artificial Intelligence in Medicine*, 6, 29—49, (1994).

Grimnes, M., Aamodt, A., A Two layer Case-based Reasoning Architecture for Medical Image Understanding. In (*Smith and Faltings 1996*), 164—178, (1996).

Grossberg, S., Levine, D., Neural Dynamics of attentionally modulated Pavlovian conditioning: blocking, interstimulus interval and secondary reinforcement. *Applied Optics*, 26, 5015—5030, (1987).

Haykin, S. Neural Networks. a comprehensive foundation. *Macmillan College Publishing Company*. Hamilton, Canada, (1999).

Hebb, D., O., The Organization of Behavior. John Wiley and Sons. New York (1949).

Heider, R. Troubleshooting CFM 56-3 engines for the Boeing 737 using CBR and data-mining, In *Proceedings of the Third European Workshop on Case-Based Reasoning*, Lausanne, Switzerland, 512—518, (1996).

Hilario, M., An overview of strategies for neurosymbolic integration. In *Connectionist Symbolic Integration*, chapter 1. Hillsdale, NJ : Lawrence Erlbaum, (1997).

Holt, A., Benwell, G.L., Applying case-based reasoning techniques to Spatial Phenomena. *In The International Journal of Geographical Information Science,* P. Fischer, Taylor and Francis (Eds.), London 13(1), 9—25, (1999).

Johnston, R., Breen, S., Manago, M., Methodology for developing CBR applications, *Technical Report,* ESPRIT Project INRECA, Deliverable D30, (1996).

Jordan, M. I., Kearns, M. J., Solla S. A. (Eds.), *Advances in Neural Information Processing Systems 10*, Proceedings of the 1997 Conference, MIT Press, (1998).

Kohonen, T., Associative Memory – A System-theoretical Approach. Springer, New York (1977).

Kohonen, T., Self-organization and Associative Memory. Springer-Verlag, Berlin (1984).

Kolodner, J. L., Case Based Reasoning. Morgan Kaufmann. (1993).

Kosko, B., Bidirectional Associative Memories. *IEEE Transactions on Systems, Man and Cybernetics*, 18, 49—60, (1988).

Leake, D.B., Kinley, A.,Wilson, D., Learning to improve case adaptation by introspective reasoning and CBR. In *Case-Based Reasoning Research and Development*, Veloso, M., Aamodt, A. (Eds.) Lecture Notes in Artificial Intelligence 1010 Springer-Verlag. Berlin. (1995).

Leake, D.B., Kinley, A., Wilson, D.C. Case-based Similarity Assessment: Estimating Adaptability from Experience. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, AAAI-97: AAAI Press. Menlo Park, CA, USA, (1997).

Leake, D.B., Wilson, D.C. Combining CBR with Interactive Knowledge Acquisition, Manipulation and Reuse. *Proceedings of the Third International Conference on Case-Based Reasoning, ICCBR-99.* (1999).

Levine, D. S. Introduction to Neural and Cognitive Modeling, Hillsdale, New Jersey, USA, Lawrence Earlbaum Associates, Inc. (1991).

Liu, Y. Calibrating an industrial microwave six-port instrument using artificial neural network technique. *IEEE Transactions in IM*, Vol. 45, No. 2, 651— 656, (1996).

Looney, C. G., Pattern Recognition Using Neural Networks, New York, USA, Oxford University Press, (1997).

Lopes, A. A., Raciocínio Baseado em Casos, Dissertação de Mestrado, ICMC-USP, São Carlos, SP (1995).

Maher, M. L., Pu P., Issues and Applications of Case-Based Reasoning in Design, Lawrence Erlbaum Associates, Mahwah, NJ, (1997).

Malek, M., Hybrid approaches for integrating neural networks and case based reasoning: from loosely coupled to tightly coupled models. In *Soft Computing in Case Based Reasoning*, chapter 4. Pal, S. K., Dillon, T. S., Yeung, D. S., eds. London, England: Springer-Verlag, (2000).

Martins, S., Computação baseada em casos: contribuições metodológicas aos modelos de indexação, de avaliação, de ranking e de similaridade de casos, Tese de Doutorado, capítulo 6, UFPB Campina Grande, (2000).

Michaud, F., Rubio, R. G. Autonomous design of artificial neural networks by neurex. In *Neural Computation*, vol 8, n. 8, 1767—1786, (1996).

Michie, D., Spiegelhalter, D.J. Taylor C.C, Machine Learning, *Neural and Statistical Classification*, Ellis Horwood, (1994).

Milaré, C. R., Carvalho, A.C.P.L.F.,: Using a Neural Network in a Case Based Reasoning System. Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, ICCIMA'98, 43—48, Vitoria, Australia, World Scientific (1998).

Netten, B.D., Vingerhoeds, R.A., Retrieval and reuse of conceptual designs in EADOCS. In *Advances in Case-Based Reasoning*, Smith and Faltings, (Eds.), (1996).

Orr, G. B., Müller, K-R., Neural networks: tricks of the trade, *Lecture Notes on Computer Science 1524*, Springer-Verlag, (1998).

Pavlov, I. P., *Conditioned Reflexes*. Oxford University Press. London (1927).

Pegler, I., Price, C. J., "Caspian: A Freeware Case-based Reasoning Shell", Proceedings of the 2nd UK Workshop on Case-Based Reasoning, (1996).

Petridis, V. and Paraschidis, K., Structural adaptation based on a simple learning algorithm. *International Joint Conference on Neural Networks*, **1**, 621—623, Baltimore, USA. (1993).

Popper, K. R., Eccles, J.C., *O Eu e seu Cérebro*, Editora UNB, Brasília, (1977).

Price, C. J. Pegler, I. S. Ratcliffe, M. B. McManus A., *From Troubleshooting to Process Design: Closing the Manufacturing Loop*, Procs 2nd Intl Conference on Case-Based Reasoning, ICCBR-97, 114—121, Providence, Rhode Island, (1997).

Quinlan, J. R., *Induction of Decision Trees*. Machine Learning, 1(1), 81—106, (1986).

Reategui, E. and Campbell, J., "A classification system for credit card transactions". *European Workshop on Case-Based Reasoning*, Keane, M., Haton, J. and Manago, M. 167—174, Chantilly, France, (1994).

Reisbeck, C. K., Schank, R. C., Inside Case-Based Reasoning, Lawrence Erlbaum Associates, (1989).

Sase, M., Matsui, K. and Kosugi, Y., Inter-generational architecture adaptation of neural networks. *International Joint Conference on Neural Networks*, **3**, 2941—2944, Baltimore, USA, (1993).

Schank, R. C., Dynamic Memory: A Theory of Learning in Computers and People, Cambridge Press, New York, (1982).

Smyth, B., McKenna, E., Modelling the Competence of Case-Bases, *Proceedings of the 4th European Workshop on Case-Based Reasoning*, EWCBR-98, Dublin, Ireland, 208—220, (1998).

Shin, C-K, and Park, S. C, Towards Integration of Memory Based Learning and Neural Networks. In *Soft Computing in Case Based Reasoning*, chapter 5. Pal, S. K., Dillon, T. S., Yeung, D. S., eds. London, England, Springer-Verlag, (2000).

Sousa, H. C., Um Framework para Criação e Simulação de Redes Neurais Artificiais utilizando Component Object Model, *Dissertação de Mestrado*, ICMC/USP, (2000).

Sovat, R. B., Carvalho, A. C. P. L. F., Um Ambiente para Desenvolvimento de Sistemas de Raciocínio Baseado em Casos. Anais do *XIX Congresso Nacional da Sociedade Brasileira de Computação*, 133—148, (1999).

Sovat, R. B., Carvalho, A. C. P. L. F., Escolha e Configuração de Modelos de Redes Neurais Artificiais Utilizando Raciocínio Baseado em Casos, Anais do *XX Congresso Nacional da Sociedade Brasileira de Computação*, (2001a).

Sovat, R. B., Carvalho, A. C. P. L. F., Retrieval and Adaptation of Cases Using an Artificial Neural Network. In *Proceedings of the Workshop of Soft Computing in Case-Based Reasoning, 4th International Conference on Case-Based Reasoning (ICCBR-01)*, Vancouver, British Columbia, Canada, (2001b).

Sovat, R. B., Carvalho, A. C. P. L. F., A Case-based Reasoning Development Environment". In *Proceedings of the International Conference in Coputational Intelligence and Multimedia Applications (ICCIMA-01),* Taikosha City, Japão, (2001c).

Sovat, R. B., Aluisio, S. M., Carvalho, A. C. P. L. F., RaBeCa : A Hybrid Case-based Reasoning Development Environment. In *Proceedings of the IX International Conference in Artificial Intelligence Tools*, (2001).

Surma, J., Vanhoof, K., Integrating Rules and Cases for the Classification Task. In *Case-Based Reasoning Research and Development.* Veloso, M., Aamodt, A., Lecture Notes in Artificial Intelligence 1010, 325—334, Springer-Verlag, Berlin, (1995)..

Tetko, I. V., Associative Neural Network, *CogPrints archive* code: cog0000144, http://cogprints.soton.ac.uk. (2001).

Thrift, P. A Neural Network model for Case-based Reasoning. *Workshop on Case-Based Reasoning*, Hammond, K. Pensacola Beach, USA, 334—337, Morgan Kaufmann, (1989).

Tversky, I. Gati, "Studies of Similarity", Cognition and Categorization, Lawrence Erlbaum Associates, 79—98, (1978).

Vargas, E.; M. Alvarez; A. de Carvalho, "Simulador Multi-threads de Redes Neurais Ontogênicas Orientado a Objetos", *Relatório Técnico No.59*, Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, São Carlos, (1997).

Watson, I. Marir, F. "Case-Based-Reasoning: A Review", *The Knowledge Review* 9, n. 4, 355—381, (1994).

Watson, I., Applying Case-Based Reasoning: Techniques for Enterprise Systems, Morgan Kaufman, (1997).

Watson, I. CBR is a methodology not a technology. In, *Research & Development in Expert Systems XV.* Miles, R., Moulton, M. Bramer, M. (Eds.), 213—223. Springer, London, (1998).

Wendel, O., Case based reasoning in a simulation environment for biological neural networks. *First European Workshop on Case-Based Reasoning.* Wess, S., Althoff, K. and Richter M. University of Kaiserslauten, Kaiserslauten, Germany, **1**, 1—5, (1993).

Weiner, J., Thurman, D. A., Mitchell, C. M., Applying case-based reasoning to aid fault management in supervisory control. In *Proceedings of the 1995 IEEE International Conference on Systems*, Man and Cybernetics, Vancouver, Canada, (1995).

Zell, A., SNNS. Stuttgart Neural Network Simulator. *User's Manual*, version 4.0 University of Stuttgart, (1995).

Zeghib, Y., Beuvron, F., Kullmann, M., Using Description Logics for Designing the Case Base in a Hybrid Approach for Diagnosis Integrating Model and Case-Based Reasoning. *Proceedings of the 4th International Conference on Case-Based Reasoning*, ICCBR 2001, Vancouver, Canada, LNAI 2080, 561—575, Springer-Verlag (2001).