

**GISELLE LOPES FERRARI**

**INTELLEC: SHELL PARA DESENVOLVIMENTO DE  
SISTEMAS ESPECIALISTAS**

**FLORIANÓPOLIS  
2005**

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**  
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA**  
**ELÉTRICA**

**INTELLEC: SHELL PARA IMPLEMENTAÇÃO DE**  
**SISTEMAS ESPECIALISTAS**

Dissertação submetida à  
Universidade Federal de Santa Catarina  
como parte dos requisitos para a  
obtenção do grau de Mestre em Engenharia Elétrica.

**GISELLE LOPES FERRARI**

Florianópolis, Fevereiro de 2005.

# INTELLEC: SHELL PARA DESENVOLVIMENTO DE SISTEMAS ESPECIALISTAS

**Giselle Lopes Ferrari**

‘Esta Dissertação foi julgada adequada para a obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em Engenharia Biomédica, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.’

---

Profa. Fernanda Isabel Marques Argoud, Dra.  
Orientador

---

Prof. Fernando Mendes de Azevedo, D.Sc.  
Co-Orientador

---

Prof. Denizar Cruz Martins, Dr.  
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

## **Banca Examinadora:**

---

Profa. Fernanda Isabel M. Argoud, Dra.  
Presidente

---

Prof. Fernando Mendes de Azevedo, D.Sc.

---

Prof. Jefferson L. Brum Marques, Ph.D.

---

Prof. Anita Maria da Rocha Fernandes, Dra.

---

Profa. Jacqueline G. Rolim, Dra.

---

Prof. Marco Aurélio B. Rodrigues, Dr.

Aos meus pais, Valdir e Rosalina.

## **AGRADECIMENTOS**

Agradeço aos meus orientadores, Fernanda Isabel Marques Argoud e Fernando Mendes de Azevedo, pela motivação e confiança para a realização deste trabalho e principalmente pela amizade.

Aos meus familiares e amigos, que mesmo estando longe, todos os dias, de alguma forma, mostraram-se sempre presentes. Obrigada pelo carinho, pela atenção, pelo amor e pelo incentivo para a realização deste.

Aos colegas do Instituto de Engenharia Biomédica, pela convivência e amizade. Em especial, a Maria Nazaré M. Angeloni Hahne, por sua constante disponibilidade para discutir idéias, problemas e soluções para o desenvolvimento deste sistema.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro.

A Deus, por tudo que tenho.

## PUBLICAÇÕES

- [1] AZEVEDO, F. M.; FERRARI, G. L.; ANGELONI, M. N. M.; ARGOUD, F. I. M.; ALGARVE, A. S. Shell para Desenvolvimento de Sistemas Especialistas Fuzzy usando Prolog. In: III Congresso Latino Americano de Engenharia Biomédica (Set. 2004: João Pessoa, Paraíba). *Anais*. Paraíba, 2004. p. 919-922.
- [2] FERRARI, G. L.; ARGOUD, F. I. M.; AZEVEDO, F. M. Shell para Desenvolvimento de Sistemas Especialistas Fuzzy – Estudo de Caso: Gastroenterologia, IV Workshop de Informática aplicada à Saúde – CBComp (Outubro 2004 : Itajaí, Santa Catarina). *Anais*. Santa Catarina, 2004. p. 583-588.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

## **INTELLEC: SHELL PARA DESENVOLVIMENTO DE SISTEMAS ESPECIALISTAS**

**Giselle Lopes Ferrari**

Fevereiro / 2005

Orientador: Fernanda Isabel Marques Argoud, Dra.  
Área de Concentração: Engenharia Biomédica  
Palavras Chave: *Shell*, Sistemas Especialistas, Prolog  
Número de Páginas: 73

Este trabalho apresenta o desenvolvimento de uma *shell*, chamada de Intellec, para a geração automática de Sistemas Especialistas, utilizando lógica *fuzzy* para tratamento da incerteza. Esta *shell* utiliza a linguagem Prolog para a implementação da máquina de inferência, por se tratar de uma linguagem de programação lógica. A lógica *fuzzy* foi escolhida para representar a incerteza a respeito do conhecimento do domínio, permitindo também o uso de variáveis lingüísticas. A interface gráfica com o usuário foi implementada utilizando-se a ferramenta Borland Delphi. A validação do sistema proposto foi feita através de comparação direta entre o Intellec e o Expert SINTA, *shell* desenvolvida pelo grupo SINTA da Universidade Federal do Ceará, a qual é amplamente utilizada e reconhecida. Dois problemas, de áreas de conhecimento bastante distintas, foram implementados simultaneamente em ambas as *shell's* e os resultados obtidos foram compatíveis.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering

# **INTELLEC: SHELL FOR THE DEVELOPMENT OF EXPERT SYSTEMS**

**Giselle Lopes Ferrari**

February / 2005

Advisor: Fernanda Isabel Marques Argoud, Dra.  
Area of Concentration: Biomedical Engineering  
KeyWords: Shell, Expert Systems, Prolog  
Number of pages:73

This work presents the development of a shell, called Intellec, for the automatic generation of Expert Systems, using fuzzy logic for the treatment of uncertainty. This shell uses the Prolog language in the implementation of the inference machine, as it is a logical programming language. The fuzzy logic was chosen to represent the uncertainty within the knowledge base, also allowing the use of linguistic variables. The graphical interface with the user was implemented using Borland Delphi. The validation of the proposed shell was made through direct comparison between the Intellec and the Expert SINTA, shell developed by the group SINTA of the Federal University of Ceará, which is widely used. Both shells were used to implement two expert systems, with similar results.



# SUMÁRIO

LISTA DE FIGURAS .....	x
LISTA DE TABELAS.....	xii
LISTA DE SIGLAS.....	xiii
1 Introdução.....	1
1.1 Justificativa do Trabalho .....	1
1.2 Sistemas Especialistas .....	2
1.2.1 Abordagem Histórica.....	2
1.2.2 Sistemas Especialistas .....	4
1.2.3 Classificação de Sistemas Especialistas .....	6
1.2.4 Características de Sistemas Especialistas .....	7
1.3 Prolog .....	9
1.4 Expert SINTA .....	11
1.5 Objetivos .....	13
1.5.1 Objetivo Geral .....	13
1.5.2 Objetivos Específicos .....	13
2 Fundamentação Teórica.....	14
2.1 Componentes de um Sistema Especialista .....	14
2.2 Estrutura de Sistemas Especialistas.....	15
2.2.1 Base de Conhecimento .....	17
2.2.2 Máquina de Inferência .....	18
2.3 Etapas na Construção de um Sistema Especialista.....	18
2.4 Métodos de Raciocínio de um Sistema Especialista .....	20
2.5 O Conhecimento.....	21
2.5.1 A Aquisição do Conhecimento.....	21
2.5.2 Métodos de Representação do Conhecimento.....	23
2.6 Tratamento de Incerteza .....	25
2.6.1 Lógica <i>Fuzzy</i> .....	26
2.6.2 Raciocínio Probabilístico.....	26

2.6.3	Fatores de Certeza .....	27
2.7	Shell.....	28
2.8	Shell para Sistemas Especialistas Fuzzy .....	29
2.8.1	Módulo Criar .....	29
2.8.2	Módulo Modificar.....	30
2.8.3	Módulo Consultar .....	31
2.8.4	Módulo Excluir.....	32
3	Implementação do Sistema.....	34
3.1	Intellec .....	34
3.1.1	Módulo Editar Base de Conhecimento.....	36
3.1.2	Módulo Consultar .....	48
3.1.3	Módulo Depurar .....	55
3.2	DBIntellec .....	57
3.2.1	Módulo Cadastro de Pacientes .....	57
3.2.2	Módulo Histórico de Consultas .....	58
4	Resultados e Discussão.....	60
4.1	Sistema Especialista em Vinhos.....	60
4.2	Sistema Especialista de Auxílio a Diagnóstico de Hepatites Virais .....	64
5	Conclusões.....	68
6	Referências Bibliográficas.....	70

## LISTA DE FIGURAS

Figura 1.1 – Engenharia de Conhecimento: transferência de conhecimento de um especialista para um programa de computador (Waterman, 1986). .....	6
Figura 1.2 – Características gerais do conhecimento embutido em um SE.....	8
Figura 1.3 – Arquitetura simplificada do Expert SINTA (NOGUEIRA et al., 1996). .....	12
Figura 2.1 – Componentes de um SE. ....	14
Figura 2.2 – Classificação de SE’S (Genaro, 1987). ....	16
Figura 2.3 –A estrutura de um SE (Waterman, 1986). ....	17
Figura 2.4 – Etapas de desenvolvimento de um SE (Waterman, 1986). ....	18
Figura 2.5 – Tela para criação do sistema. ....	30
Figura 2.6 – Tela para gerenciamento das questões .....	31
Figura 2.7 – Tela de consulta do sistema.....	32
Figura 2.8 – Tela para exclusão de um especialista. ....	33
Figura 3.1 – Esquema de funcionamento do Intellec. ....	35
Figura 3.2 - Relacionamento entre os módulos e DLLs do sistema.....	36
Figura 3.3 – Descrição do Sistema Especialista. ....	37
Figura 3.4 – Tela de edição de variáveis. ....	38
Figura 3.5 – Pergunta utilizando variável do tipo imagem. As imagens utilizadas foram retiradas do endereço <a href="http://www.turmadapele.com.br/dicas/fotoprotecao_cancer.asp">http://www.turmadapele.com.br/dicas/fotoprotecao_cancer.asp</a> , acessado no dia 10/12/2004.....	39
Figura 3.6 – Variável. ....	39
Figura 3.7 – Objetivos. ....	40
Figura 3.8 – Tela de edição de regras. ....	41
Figura 3.9 – Regras.....	42
Figura 3.10 – Tela de edição de valores de verdade.....	43
Figura 3.11 – Valor verdade. ....	44
Figura 3.12 – Tela de edição de perguntas. ....	45
Figura 3.13 – Perguntas. ....	46
Figura 3.14 – Tela de consulta do sistema.....	50
Figura 3.15 – Pergunta fuzzy.....	54
Figura 3.16 – Pergunta crisp.....	55

Figura 3.17 – Regras aceitas pelo sistema.....	56
Figura 3.18 – Valores das variáveis.....	56
Figura 3.19 – Informações sobre o paciente.....	58
Figura 3.20 – Histórico das consultas.....	59

## LISTA DE TABELAS

Tabela 4.1 – Resultado obtido para a primeira variável objetivo – cor recomendada. ....	61
Tabela 4.2 -- Resultado obtido para a segunda variável objetivo – doçura recomendada..	61
Tabela 4.3 – Resultado obtido para a terceira variável objetivo – vinho. ....	61
Tabela 4.4 – Resultado obtido para a primeira variável objetivo – cor recomendada. ....	63
Tabela 4.5 – Resultado obtido para a segunda variável objetivo – doçura recomendada..	63
Tabela 4.6 – Resultado obtido para a terceira variável objetivo – vinho. ....	63
Tabela 4.7– Resultado obtido para a primeira variável objetivo – sugestivo de hepatite viral.....	66
Tabela 4.8– Resultado obtido para a primeira variável objetivo – possibilidade diagnóstica final. ....	66
Tabela 4.9 – Resultado obtido para a primeira variável objetivo – sugestivo de hepatite viral. ....	67
Tabela 4.10 – Resultado obtido para a primeira variável objetivo – possibilidade diagnóstica final. ....	67

## LISTA DE SIGLAS

ALT – Alanina aminotransferase  
Anti-HbcIgM – Anticorpo para antígeno de centro do vírus de hepatite B  
Anti-HBs – Anticorpo para antígeno de superfície do vírus de hepatite B  
Anti-HCV – Anticorpo para o vírus de hepatite C  
BC – Base de Conhecimento  
CCS – Centro de Ciências da Saúde  
EC – Engenharia de Conhecimento  
FC – Fator de Certeza  
HBeAg – Antígeno “e” da hepatite B  
HBsAg – Antígeno de superfície do vírus de hepatite B  
HBV-DNA – Pesquisa do ácido desoxirribonucleico do vírus de hepatite B  
HCV-RNA – Pesquisa do ácido ribonucleico do vírus de hepatite C  
HTML – Hypertext Markup Language  
IEB-UFSC – Instituto de Engenharia Biomédica  
IA – Inteligência Artificial  
LIA – Laboratório de Inteligência Artificial  
MC – Medida de Crença  
MD – Medida de Descrença  
MI – Máquina de Inferência  
MIT – Massachusetts Institute of Technology  
PDF – Portable Document Format  
SAFO – Sistema Automático para Formalização do Conhecimento  
SBC – Sistema Baseado em Conhecimento  
SE – Sistema Especialista  
SINTA – Sistemas Inteligentes Aplicados  
SP – Sistemas de Produção  
SSEF – Shell para Sistemas Especialistas Fuzzy  
UFSC – Universidade Federal de Santa Catarina

# 1 Introdução

## 1.1 Justificativa do Trabalho

Nos últimos anos tem crescido bastante o número de trabalhos e pesquisas de ferramentas computacionais em Inteligência Artificial, que procuram capturar e simular o comportamento de especialistas humanos.

Os Sistemas Especialistas despertam grande interesse, principalmente pelo fato de serem um dos produtos mais viáveis e aplicados nos estudos em Inteligência Artificial. Eles têm sido utilizados em uma série de aplicações, como: sistemas de auxílio ao diagnóstico; sistemas de predição; sistemas de reconhecimento de padrões e sistemas tutoriais inteligentes.

Porém, a construção de um software para a geração automática de sistemas especialistas - *shell* - não é trivial, tendo em vista que o mesmo deve ser capaz de gerar sistemas que se propõem a tratar problemas complexos do mundo real, que necessitam da interpretação de um especialista, precisam chegar simultaneamente às mesmas conclusões a que chegaria o especialista humano, caso se defrontasse com problemas equivalentes. A importância da utilização de Sistemas Especialistas deve-se a diversos fatores tecnológicos e econômico-sociais, dentre os quais tem-se: a dificuldade de acesso a especialistas humanos em determinadas regiões, a possibilidade de armazenamento e formalização do conhecimento de vários especialistas humanos, a disponibilização de uma ferramenta de apoio à tomada de decisões por parte do especialista, para auxiliar o treinamento de profissionais e pela imparcialidade na tomada de decisões (NOGUEIRA *et al.*, 1996).

A motivação para o desenvolvimento de uma *shell* surgiu de dificuldades (como por exemplo, tratamento de imagens, banco de dados para registrar as consultas realizadas e a utilização de tutoriais para deixar os sistemas especialistas mais didáticos) encontradas em vários projetos desenvolvidos no Instituto de Engenharia Biomédica (IEB-UFSC),

tanto em trabalhos de Dissertação de Mestrado como em Teses de Doutorado, na área de Informática Médica, que utilizam Sistemas Especialistas para auxílio ao diagnóstico em diferentes áreas da medicina. Ressalta-se também o interesse demonstrado pelos alunos do Curso de Mestrado em Medicina, do Centro de Ciências da Saúde (CCS), da Universidade Federal de Santa Catarina (UFSC), do qual o IEB-UFSC participa.

Em todos os casos, uma solução encontrada pelos pesquisadores do IEB-UFSC foi a utilização do Expert SINTA, uma *shell* desenvolvida no Laboratório de Inteligência Artificial da Universidade Federal do Ceará. Este grupo de pesquisa descontinuou os desenvolvimentos necessários ao Expert SINTA para torná-lo mais viável e aplicável a problemas reais. Logo, tornou-se necessário o desenvolvimento de um outro sistema com características semelhantes ou melhores que as do Expert SINTA, como uma alternativa aos caros sistemas comercializados nos Estados Unidos. Como exemplos dessas *shells*, pode-se citar o NETICA (NORSYS SOFTWARE CORP., 2005) e o ACQUIRE (ACQUIRED INTELLIGENCE INC., 2005).

## **1.2 Sistemas Especialistas**

### **1.2.1 Abordagem Histórica**

A história da Inteligência Artificial (IA) transcende bastante a do computador atual. Nos anos 50, os pesquisadores já haviam estabelecido as fundações da IA, incluindo lógica matemática e teoria das funções recursivas. Aproximadamente nesta mesma ocasião, psicólogos cognitivos – pesquisadores da forma de pensar humana – criaram caminhos, padrões do processo de investigação do raciocínio, modelando o aparente processo de tomada de decisão, em termos de regras de produção condicionais.

No início da década de 1960, começaram os primeiros trabalhos nos sistemas que hoje são chamados de “especialistas”. Inicialmente, pretendia-se construir máquinas inteligentes, com grande poder de raciocínio e resolução de problemas. Imaginava-se que, a partir de um pequeno conjunto de normas ou regras de raciocínio introduzidas num poderoso computador criar-se-iam sistemas de capacidade superior à humana. Não tardou



para que os pesquisadores observassem o engano e verificassem as reais dimensões do trabalho (HEINZLE, 1995).

Em 1964 foi construído o DENDRAL, por Joshua Lederberg, da Universidade de Stanford (EUA), um sistema capaz de inferir a estrutura molecular de compostos desconhecidos, a partir de dados espectrais de massa e resposta magnética nuclear (BITTENCOURT, 2001). Em 1965, Lederberg juntou-se a Edward Feigenbaum e Bruce Buchanan, com o intuito de desenvolver um sistema que empregasse heurísticas para resolver os mesmos problemas do DENDRAL (HARMON, 1988). Surge então a idéia fundamental dos sistemas especialistas - a engenharia do conhecimento.

Em 1968, surge no MIT – Massachusetts Institute of Technology – USA, o MACSYMA, destinado a auxiliar matemáticos na resolução de problemas complexos. O MACSYMA é um sistema ainda hoje amplamente utilizado em Universidades e laboratórios de pesquisa.

Posteriormente, já na década de 1970, surgiram importantes e complexos sistemas especialistas, dentre os quais destacam-se: MYCIN, CADUCEUS, EXPERT, SOPHIE e o PROSPECTOR.

O MYCIN, desenvolvido na Universidade de Stanford, EUA (1975), é um sistema que auxilia médicos na escolha de uma terapia de antibióticos para pacientes com bacteremia, meningite e cistite infecciosa, em ambiente hospitalar (BITTENCOURT, 2001).

O PROSPECTOR, estruturado de forma similar ao MYCIN, é um sistema para dar suporte a geólogos na exploração mineral.

A década de 1980 foi marcada pelo grande crescimento de aplicações, inclusive com larga disponibilização de produtos comerciais no mercado de software. Este acelerado processo de desenvolvimento de aplicações deve-se em parte ao avanço dos recursos de equipamentos, ou hardware, ocorrida paralelamente neste período (HEINZLE, 1995).

No Brasil, a Pontifícia Universidade Católica do Rio de Janeiro desenvolveu nos anos 80, importantes trabalhos com sistemas especialistas. O principal resultado obtido nesta área é um sistema chamado SAFO (Sistema Automático para Formalização do Conhecimento), cuja finalidade é a demonstração de teoremas matemáticos. O Instituto Militar de Engenharia foi o pioneiro em pesquisas em IA no Brasil, já tendo desenvolvido algumas ferramentas para aplicação de técnicas de IA (MONTELLO, 1999).

## 1.2.2 Sistemas Especialistas

“Um Sistema Especialista (SE) é aquele que é projetado e desenvolvido para atender uma aplicação determinada e limitada do conhecimento humano. É capaz de emitir uma decisão, com apoio em conhecimento justificado, a partir de uma base de informações, tal qual um especialista de determinada área do conhecimento humano” (CUNHA e RIBEIRO, 1987).

Os SEs são sistemas computacionais projetados e desenvolvidos com o objetivo de solucionar problemas de uma forma semelhante àquela utilizada pelo especialista do domínio. O especialista é aquela pessoa que, através de treinamento e experiência, alcançou um alto grau de conhecimento e competência em uma determinada área do conhecimento humano. Ele é capaz de executar coisas que os outros não conseguem; são exímios e eficientes no que fazem (*apud* MONTELLO, 1999).

O conhecimento contido numa especialidade pode ser:

- Conhecimento Público: As definições públicas de fatos e teorias, tais como livros-texto e referências, nos quais o domínio do estudo está representado.

- Conhecimento Privado: Conhecimento que o especialista foi adquirindo mediante a experiência pessoal na área e que não é publicado na literatura. O conhecimento privado consiste em um grande número de regras intuitivas heurísticas (*apud* VIERA, 1996).

Para alcançar o desempenho de especialistas humanos, o SE deve possuir, não só um conjunto de informações, mas também a habilidade de utilizá-las na resolução de problemas de forma criativa, correta e eficiente. Esta habilidade representa uma série de

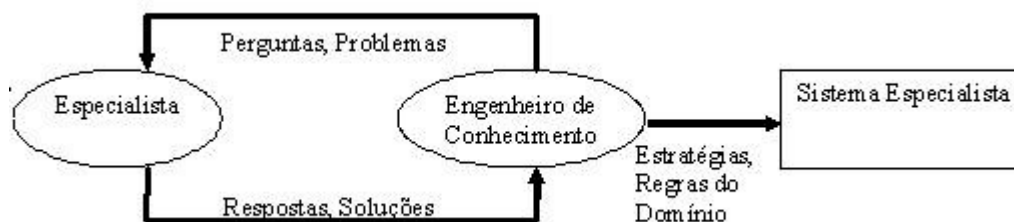
regras intuitivas que o especialista utiliza para resolver os problemas, e sua aplicação possibilita, de uma maneira mais econômica, a obtenção de soluções aceitáveis, embora nem sempre ótimas (CUNHA, 1995).

Para que um sistema especialista seja uma ferramenta eficaz, as pessoas deverão ser capazes de interagir com o mesmo facilmente. Para facilitar esta interação, é importante que um sistema especialista, além da capacidade de realizar a sua tarefa, tenha as duas habilidades seguintes (RICH, 1988):

- A explicação do seu raciocínio. Em muitos dos domínios onde os sistemas especialistas operam, as pessoas não aceitarão resultados a menos que estejam convencidas da precisão do processo de raciocínio que produziu esses resultados. Isto é particularmente verdadeiro, por exemplo, na medicina, pois o médico terá a responsabilidade final pelo diagnóstico, mesmo quando este for alcançado com considerável auxílio de um programa. Assim, é importante que o processo de raciocínio utilizado nesses programas possua passos compreensíveis e que o meta-conhecimento (conhecimento a respeito do processo de raciocínio) suficiente esteja disponível, para que as explicações desses passos possam ser geradas.

- A aquisição de conhecimento. Como os sistemas especialistas derivam seu poder da riqueza das bases de conhecimento que exploram, é extremamente importante que essas bases sejam tão completas e precisas quanto possível. Muitas vezes, entretanto, não existe nenhuma codificação padrão desse conhecimento; ou seja, ele só existe dentro da cabeça do especialista humano. Assim, a única maneira de levar este conhecimento para o programa será pela interação com o especialista humano.

O processo de construção de um SE é frequentemente chamado de Engenharia de Conhecimento (EC). Tipicamente envolve uma forma especial de interação entre o construtor do SE, chamado Engenheiro de Conhecimento, e um ou mais especialistas em alguma área. O Engenheiro de Conhecimento “extrai” dos especialistas seus procedimentos, estratégias e regras práticas para a solução de problemas e reconstrói este conhecimento em um SE, como mostra a Figura 1.1. O resultado é um programa que soluciona problemas à maneira dos especialistas humanos (WATERMAN, 1986).



**Figura 1.1 – Engenharia de Conhecimento: transferência de conhecimento de um especialista para um programa de computador (Waterman, 1986).**

### 1.2.3 Classificação de Sistemas Especialistas

De acordo com as características de funcionamento, os SEs podem ser classificados nas categorias abaixo (FERNANDES, 2004):

- Interpretação: Sistemas que inferem descrições de situações a partir da observação de fatos, fazendo uma análise de dados e procurando determinar suas relações e seus significados.

- Diagnósticos: Sistemas que detectam falhas oriundas da interpretação de dados.

- Monitoramento: Sistemas que devem verificar, de maneira contínua, um determinado comportamento em limites preestabelecidos, sinalizando quando forem requeridas intervenções para o sucesso da execução.

- Predição: A partir de uma modelagem de dados históricos e atuais, este sistema permite uma determinada previsão do futuro.

- Planejamento: O sistema prepara um programa de iniciativas a serem tomadas para se atingir um determinado objetivo.

- Projeto: Tem características parecidas com as do planejamento; é um sistema capaz de justificar a alternativa tomada para o projeto final, e de fazer uso dessa justificativa para alternativas futuras.

- Depuração: Sistema que possui mecanismos para fornecer soluções para o mau funcionamento provocado por distorções de dados.

- Reparo: Este sistema desenvolve e executa planos para administrar os reparos verificados na etapa de diagnóstico.

- Instrução: Tem mecanismos para verificar e corrigir o comportamento do aprendizado dos estudantes.

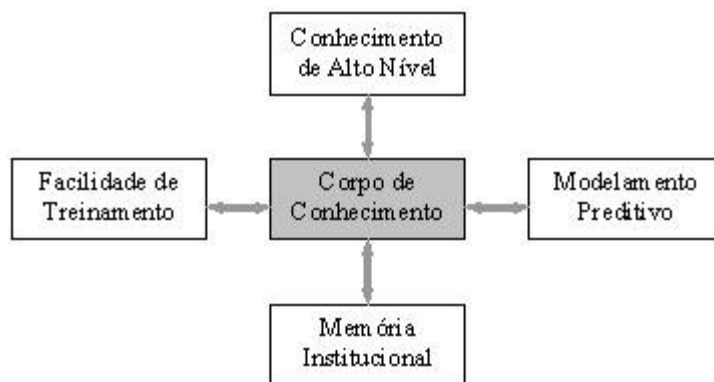
- Controle: É um sistema que governa o comportamento geral de outros sistemas (não apenas de computação). Deve interpretar os fatos de uma situação atual, verificando os dados passados e fazendo uma predição do futuro.

#### **1.2.4 Características de Sistemas Especialistas**

As características gerais do conhecimento embutido em um Sistema Especialista são: corpo do conhecimento, conhecimento de alto nível, modelamento preditivo, memória institucional e facilidade de treinamento.

##### ➤ Corpo de Conhecimento

O conhecimento adquirido durante a construção de um SE deve ser explícito e organizado para simplificar a tomada de decisão. “A aquisição e codificação de conhecimento é um dos mais importantes aspectos de um sistema especialista” (WATERMAN, 1986). A Figura 1.2 apresenta as características principais de um SE.



**Figura 1.2 – Características gerais do conhecimento embutido em um SE.**

➤ **Conhecimento de Alto Nível**

É uma das características mais utilizadas, que permite ao SE atuar como um especialista e produzir resultados de alta qualidade num tempo mínimo, provendo ajuda na resolução de problemas. Isto se deve ao fato que o conhecimento representado no SE é a opinião de especialistas de alto nível, o que representa anos de experiência em uma determinada tarefa (WATERMAN, 1986).

➤ **Modelamento Preditivo**

O sistema pode agir como uma teoria ou modelo de processamento de informação de solução de problema em um dado domínio, fornecendo as respostas desejadas de um determinado problema e mostrando como ele se comportaria para novas situações (WATERMAN, 1986).

➤ **Memória Institucional**

O Corpo de Conhecimento que define a proficiência de um sistema especialista pode também oferecer uma característica adicional, uma memória institucional. Se a base de conhecimento foi desenvolvida através de interações de pessoas chaves num escritório ou departamento, representa a política atual ou procedimento operacional de tal grupo. Esta compilação de conhecimentos apresenta um consenso de opiniões de alto nível e um registro permanente das melhores estratégias e os métodos utilizados pelo grupo de

trabalho. Quando as pessoas chaves vão embora, ainda fica retido seu conhecimento (WATERMAN, 1986).

➤ Facilidade de Treinamento

Um SE é capaz de oferecer treinamento, uma vez que já contém conhecimento e habilidade de explicar seu processo de raciocínio (GENARO, 1987).

### **1.3 Prolog**

A Prolog (*Programming Logic*), como uma linguagem declarativa, oferece muitas características que facilitam a implementação de programas de IA, principalmente na área de Sistemas Especialistas.

Foi inventada por volta de 1970, por Alain Colmerauer na França, porém o interesse pela Prolog realmente cresceu depois que os japoneses lançaram o projeto da Quinta Geração de Computadores.

É uma linguagem declarativa, isto é, fornece ao programa fatos e relacionamentos (por exemplo, dados e base de conhecimento) de um determinado problema, e a solução é encontrada pela máquina de inferência interna. Diferente acontece com as linguagens procedimentais, nas quais é necessário especificar precisamente todos os passos a serem executados pelo programa (IGNIZIO, 1991).

O conjunto de declarações é também chamado de “base de conhecimento para Prolog”. A base de conhecimento é composta por fatos e regras e serve para determinar se uma dada pergunta feita pelo usuário é ou não relevante para a interpretação. A Prolog usa sua base de conhecimento e aplica a sua regra de inferência, utilizando a Lógica de Predicados e do Princípio de Resolução para resolver um determinado problema.

A Prolog é uma representação lógica de primeira ordem, ou seja, representação que permite qualificação sobre os indivíduos, mas não sobre os predicados. A representação em

lógica de primeira ordem provém de uma sintaxe bem formada, uma semântica clara e, acima de tudo, as noções de verdade e inferência. Além disso, ela exhibe as seguintes vantagens sobre outros sistemas de representação (ARARIBÓIA, 1988):

- A Lógica freqüentemente apresenta-se como uma maneira natural para expressar certas noções. A representação de um problema em Lógica corresponde, na maioria das vezes, ao entendimento intuitivo que se tem do domínio do problema.

- A Lógica é precisa. O vínculo semântico de um conjunto de declarações lógicas (isto é, o conjunto de inferências ou conclusões que podem ser delineadas a partir das declarações) é completamente especificado pelas regras de inferência. Teoricamente, a base de fatos ou informações pode ser mantida logicamente consistente e todas as conclusões são garantidamente corretas.

- A Lógica é flexível. Os fatos ou informações estão representados de uma forma que permite uma interpretação global e o mesmo fato pode ser usado para múltiplas finalidades.

- A Lógica é modular. Outras declarações lógicas podem ser colocadas na base de conhecimento, independentemente das já existentes. O conhecimento do sistema pode crescer à medida que novos fatos são descobertos e adicionados à base.

A máquina de inferência da Prolog utiliza como estratégia de raciocínio o encadeamento regressivo e *backtracking*, ou seja, quando falha a tentativa de busca de uma solução, ele procura um outro caminho para tentar satisfazer o objetivo (IGNIZIO, 1991).

As principais características da Prolog são (BARRETO, 1997):

- Programa e dados têm a mesma estrutura;
- *Backtracking*, que permite achar múltiplas soluções;
- Favorece o paradigma da programação em Lógica;



- Devido à sua estrutura quase declarativa, é adequada ao uso do paralelismo;
- Sua concisão e conseqüente curto tempo de desenvolvimento fazem da Prolog uma linguagem útil de prototipagem;
- Prolog, sendo uma linguagem de alto nível, requer muitos recursos computacionais, motivando avanços no *hardware* de computadores.

#### **1.4 Expert SINTA**

O Expert SINTA foi desenvolvido pelo grupo SINTA (Sistemas Inteligentes Aplicados), integrante do Laboratório de Inteligência Artificial (LIA), do Departamento de Computação da Universidade Federal do Ceará, Brasil.

O Expert SINTA é uma ferramenta computacional que utiliza técnicas de Inteligência Artificial para a geração automática de sistemas especialistas. Esta ferramenta utiliza um modelo de representação do conhecimento baseado em regras de produção e fatores de certeza, tendo como objetivo principal simplificar o trabalho de implementação de sistemas especialistas através do uso de uma máquina de inferência compartilhada, da construção automática de telas e menus, do tratamento probabilístico das regras de produção e da utilização de explicações sensíveis ao contexto da base de conhecimento modelada (NOGUEIRA *et al.*, 1996).

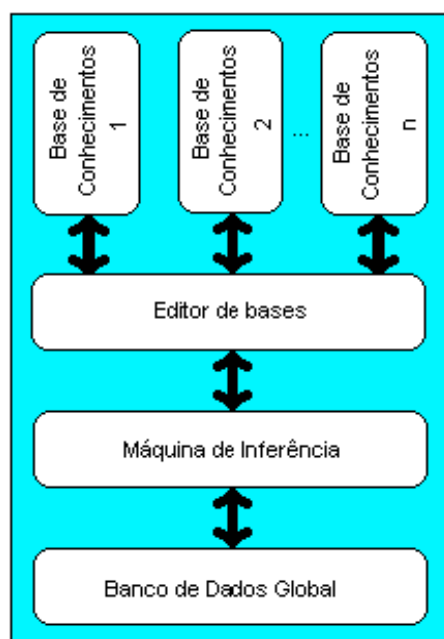
Um sistema especialista baseado neste tipo de modelo é bastante útil em problemas de classificação. O usuário responde a uma seqüência de menus e o sistema encarrega-se de fornecer respostas que se encaixem no quadro apontado pelo usuário.

Entre outras características inerentes ao Expert SINTA, destacam-se:

- Utilização do encadeamento regressivo;

- Utilização de fatores de certeza;
- Ferramentas de depuração;
- Possibilidade de incluir ajudas *on-line* para cada base.

Segundo Nogueira *et al.* (1996), os Sistemas Especialistas gerados no Expert SINTA seguem a arquitetura apresentada na Figura 1.3.



**Figura 1.3 – Arquitetura simplificada do Expert SINTA (NOGUEIRA *et al.*, 1996).**

Onde:

- A base de conhecimentos: representa computacionalmente a informação que o especialista utiliza;
- O editor de bases: é o meio pelo qual a *shell* permite a implementação das bases desejadas;

- A máquina de inferência: é a parte do SE responsável pelas deduções sobre a base de conhecimentos;

- O banco de dados global: são as evidências apontadas pelo usuário do SE, durante a consulta.

## **1.5 Objetivos**

### **1.5.1 Objetivo Geral**

Implementar uma *shell* para desenvolvimento de Sistemas Especialistas, utilizando a máquina de inferência implementada por Hahne (2001).

### **1.5.2 Objetivos Específicos**

- Implementar uma *shell* que possa permitir ao próprio especialista de domínio desenvolver um sistema;

- Permitir a construção de Sistemas Especialistas com várias metas;

- Criar imagens como “valores” de variáveis;

- Criar tutoriais, como explicações para as perguntas;

- Criar cadastro de pacientes e histórico de consultas e

- Implementar estudo de casos.

## 2 Fundamentação Teórica

### 2.1 Componentes de um Sistema Especialista

A Figura 2.1 ilustra os elementos mais importantes na construção de um SE: o SE em si, o especialista no domínio, o engenheiro do conhecimento, a ferramenta de construção do SE e o usuário.

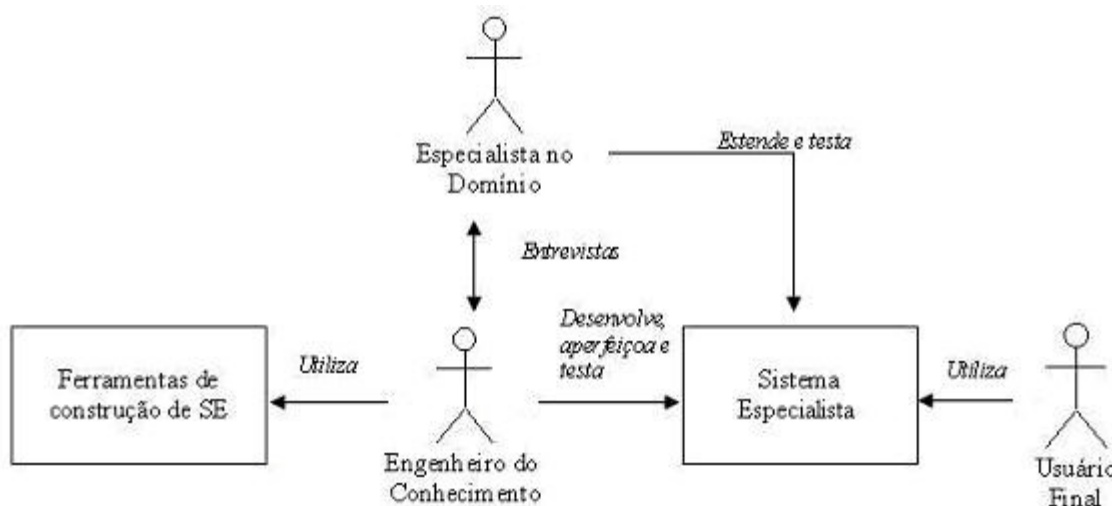


Figura 2.1 – Componentes de um SE.

#### ➤ Sistema Especialista

Os Sistemas Especialistas são sistemas computacionais que devem apresentar um comportamento semelhante a um especialista em um determinado domínio (BARRETO, 1997). Um sistema especialista deve ser construído com o auxílio de um especialista humano, o qual fornecerá a base de informações, através de seu conhecimento e experiências adquiridos ao longo dos anos (FERNANDES, 2004).

#### ➤ Especialista no Domínio

O Especialista no Domínio é uma pessoa informada e reconhecida por produzir soluções ótimas, a problemas particulares, de uma determinada área. É aquela pessoa que,

através de muito treinamento, estudo e experiência, adquiriu um alto nível de desenvolvimento em uma determinada área do conhecimento humano. O especialista é a fonte de conhecimento do SE.

➤ Engenheiro do Conhecimento

O Engenheiro do Conhecimento é o responsável pela criação da base de conhecimento. Ele obtém o conhecimento do especialista de domínio, com o uso de técnicas de elicitación de conhecimento. O conhecimento é transformado de forma conveniente e é adicionado à base de conhecimento. O Engenheiro de Conhecimento também é responsável pela escolha das ferramentas de construção mais apropriada.

➤ A Ferramenta de Construção do SE

A Ferramenta de Construção do SE é a linguagem de programação ou ambiente de desenvolvimento (*shells*) utilizado pelo *Engenheiro do Conhecimento* ou programadores, para a construção do SE (WATERMAN, 1986).

➤ Usuário Final

O Usuário Final é a pessoa que utiliza o SE, podendo ser outro especialista, um aprendiz ou um leigo, naquela área de conhecimento.

## **2.2 Estrutura de Sistemas Especialistas**

Pensando na resolução de problemas utilizando os princípios de IA, a primeira preocupação deve ser com a definição do “conhecimento de domínio do problema”, pois em qualquer área de IA, a técnica para a resolução de problemas baseia-se neste conhecimento. Uma grande quantidade de conhecimento é tão crítica quanto o é a questão de representação deste conhecimento (RICH, 1988).

Waterman (1986) classifica os SEs como Sistemas Baseados em Conhecimentos, entretanto, a recíproca não é verdadeira, isto é, nem todos os Sistemas Baseados em Conhecimento podem ser classificados em SEs, mas todos podem ser considerados como pertencentes à grande área da IA. A Figura 2.2 apresenta a classificação dos SEs, dentro do campo maior da IA, onde:

- Programas de Inteligência Artificial são programas que exigem comportamento inteligente através da aplicação apropriada de heurísticas.

- Sistemas Baseados em Conhecimento são sistemas onde o domínio do conhecimento é explícito e separado do restante do sistema.

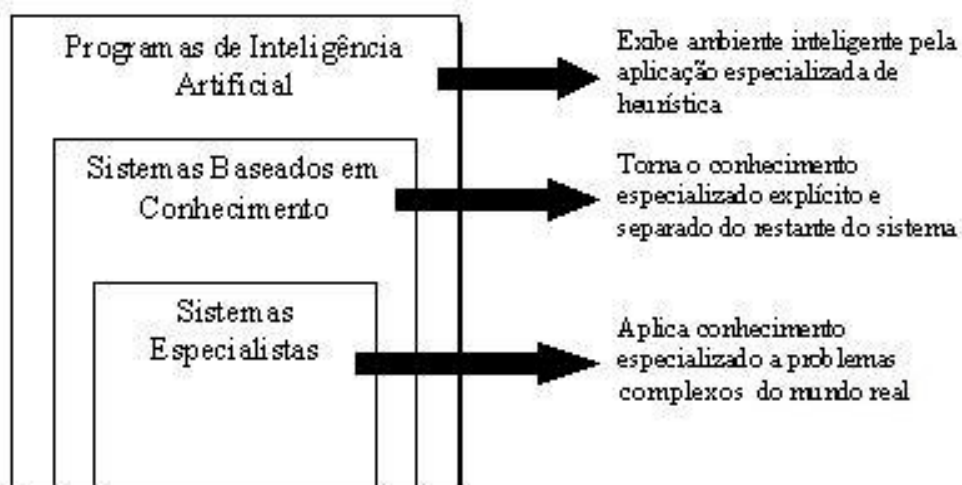


Figura 2.2 – Classificação de SE'S (Genaro, 1987).

Quando cientistas de IA usam o termo “conhecimento”, eles se referem à informação que um programa precisa possuir para que possa se comportar de maneira inteligente. O conhecimento em um SE é organizado de maneira a separar o conhecimento sobre o problema do conhecimento de sistemas, tais como conhecimentos gerais sobre como solucionar problemas ou como interagir com o usuário. Esta coleção de conhecimentos sobre o domínio do problema é chamada Base de Conhecimento (BC), enquanto o conhecimento geral para a solução do problema é chamado Motor ou Máquina de Inferência (MI). Um programa com o conhecimento organizado desta forma é chamado

de Sistema Baseado em Conhecimento (SBC) (WATERMAN, 1986). A estrutura de um SE pode ser visualizada na Figura 2.3.

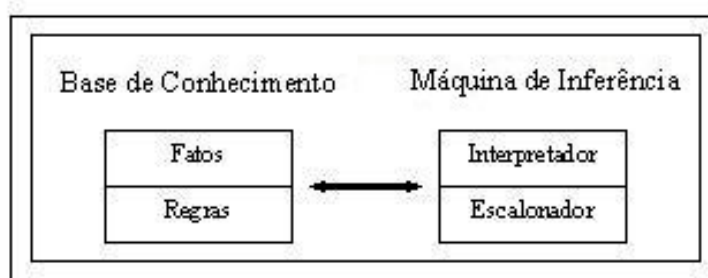


Figura 2.3 –A estrutura de um SE (Waterman, 1986).

### 2.2.1 Base de Conhecimento

A Base de Conhecimento é o conjunto de conhecimentos a respeito do domínio do problema que será utilizado nas tomadas de decisão, feitas através de fatos e regras, ou outro tipo de representação, tal como lógica matemática, redes semânticas ou “frames”.

- A Base de Fatos representa os conhecimentos que são, “a priori”, conhecidos e que podem ser considerados como o ponto de partida para a resolução do problema. São também caracterizados como o conhecimento de domínio público, de fácil acesso e que podem ser extraídos através de textos, manuais, normas, livros, constatação de fatos e resultados de experimentos (MONTELLO, 1999).

- A Base de Regras representa os conhecimentos que são extraídos diretamente dos especialistas. Estes conhecimentos representam o “pensamento” desenvolvido pelo especialista (“heurística”), tendo por base os fatos já conhecidos e as deduções a partir deles. Aqui, o termo “heurística” significa a habilidade, ou a simplificação utilizada pelo especialista, no sentido de otimizar a busca da solução de um problema. Desta forma, novos conhecimentos podem ser acrescentados à base de conhecimentos, habilitando o SE a uma tomada de decisão sobre o problema (MONTELLO, 1999).

## 2.2.2 Máquina de Inferência

A Máquina de Inferência (MI) contém um interpretador que decide como aplicar as regras para inferir novo conhecimento, além de uma lista de prioridade de aplicação destas regras (WATERMAN,1986). Nele são implementados modos de raciocínio, estratégias de busca, resolução de conflito, representação de incerteza e de conhecimento. Basicamente, a MI desempenha duas funções principais:

- Interpretador/Inferência: A partir dos conhecimentos contidos na base de conhecimento e na memória de trabalho, a MI determina quais as regras que devem ser disparadas para inferir novos conhecimentos.

- Programador/Controle: A MI determina ou programa a ordem em que as regras devem ser aplicadas.

## 2.3 Etapas na Construção de um Sistema Especialista

O desenvolvimento de um SE é realizado a partir das cinco fases interdependentes e superpostas: Identificação, Conceituação, Formalização, Implementação, Teste e Avaliação (Figura 2.4).

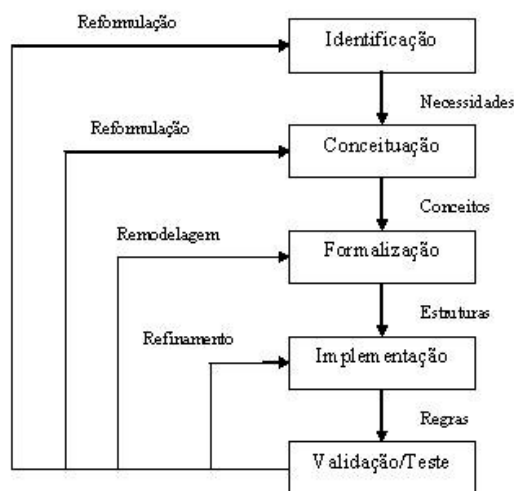


Figura 2.4 – Etapas de desenvolvimento de um SE (Waterman, 1986).



➤ Identificação

Nesta etapa são identificadas as características básicas do problema a ser resolvido. Isto inclui identificação dos participantes do projeto (engenheiro do conhecimento e especialista de domínio), dos recursos envolvidos (fontes do conhecimento, cronograma, recursos computacionais e financeiros), das características do problema (caracterização ou definição do problema e a definição dos dados) e das metas e objetivos para a construção do SE.

➤ Conceituação

Na etapa de conceituação, o engenheiro do conhecimento e o especialista de domínio determinam os conceitos, relações e mecanismos de controle que são necessários para descrever o problema a ser solucionado e estabelecer o nível de detalhamento que será usado na representação do conhecimento. Nesta fase não é feita uma análise completa do problema, pois após a implementação do protótipo, certamente ela será retomada.

➤ Formalização

A etapa de formalização envolve a expressão de conceitos e de relações-chaves, de uma maneira formal. O engenheiro do conhecimento deve prender sua atenção em três aspectos: o espaço de hipóteses (refinamento dos conceitos, características e interligação), o modelo subjacente (como as soluções serão geradas) e as características dos dados (definição de tipos, precisão, consistência, volume e formas de aquisição).

➤ Implementação

Antes de se partir para a implementação definitiva do SE, é necessário fazer um protótipo onde serão ensaiadas as várias formas de desempenho e onde deverão ser feitos os testes, verificando o desempenho. Após a homologação do protótipo, então se parte para a implementação final do sistema.

Na fase de implementação, o especialista deve levantar os conceitos e regras de sua área de conhecimento, segundo as estratégias e hierarquias definidas na fase de formalização para que o engenheiro do conhecimento possa adaptá-las às estruturas determinadas previamente e de acordo com a linguagem de programação a ser utilizada para a implementação.

➤ Testes e Avaliação

O SE deve ser testado e avaliado freqüentemente, desde a implementação do protótipo inicial, levando-se em consideração o desempenho e a utilidade. Algumas das perguntas que o especialista deve fazer na hora de testar o desempenho do protótipo são (WATERMAN, 1986):

- O sistema toma decisões que o especialista concorda como apropriadas?
- As regras de inferências são corretas, consistentes e completas?
- As estratégias de controle permitirão que o sistema considere os itens numa ordem natural, como o especialista prefere?
- As fundamentações do sistema são adequadas para descrever como e porque uma conclusão foi alcançada?

## **2.4 Métodos de Raciocínio de um Sistema Especialista**

Os SEs geralmente optam por uma das seguintes estratégias de raciocínio:

- Encadeamento Progressivo (*Forward Chaining*): O sistema é dirigido pelos dados, parte de fatos conhecidos e tenta deduzir novos fatos, através da MI, até chegar à solução (HEINZLE, 1995). Também é conhecido como encadeamento dirigido por dados.

- Encadeamento Regressivo (*Backward Chaining*): O sistema faz o caminho inverso, partindo da solução do problema (meta), a qual tenta verificar se é verdadeira através de suas condições, que passam a ser então submetidas também à aprovação. Isto ocorre sucessivamente, até se chegar a um conjunto de condições verificáveis (HEINZLE, 1995). Também é conhecido como encadeamento dirigido por objetivos.

- Encadeamento Misto: Os encadeamentos progressivos e regressivos se alternam de acordo com o desenvolvimento da solução do problema e com a disponibilidade de dados (BITTENCOURT, 2001).

## **2.5 O Conhecimento**

O conhecimento pode ser definido como a informação armazenada, ou os modelos usados pela pessoa ou máquina para interpretar, prever e responder apropriadamente ao mundo exterior. É um conceito que escapa ao modelo puramente biológico, ainda não totalmente entendido, abrindo espaço para as idéias já plenamente aceitas e usadas, especialmente pela comunidade “computacional” (FISCHLER e FIRSCHEIN, 1987).

### **2.5.1 A Aquisição do Conhecimento**

As pessoas, desde o nascimento, vão acumulando conhecimentos que lhes permitirão agir de modo a mostrar que são seres inteligentes. Isto é denominado de “aprendizado”. Geralmente pensa-se sobre o aprendizado de forma muito simplificada, como a acumulação de conhecimento. Porém, sabe-se que o aprendizado envolve um ciclo completo de processamento da informação, que vai desde a coleta do conhecimento pelos sentidos, até seu armazenamento definitivo no cérebro. Além disso, uma pessoa não capta todo o conhecimento que a realidade do mundo lhe proporciona, nem acumula definitivamente toda a informação percebida. Na verdade, o aprendizado varia de acordo com cada indivíduo (RABUSKE, 1998).

Algumas das técnicas mais utilizadas para a extração do conhecimento ou elicitação são (*apud* BRASIL, 1994):

- Observação: O especialista é observado durante seu trabalho, proporcionando uma visão realista de como trabalha. Este é freqüentemente o primeiro passo na construção de uma base de conhecimentos.

- Entrevista com o especialista: Deve ser abordada quando o elicitador já tem alguma familiaridade com o assunto. Uma entrevista deve ser planejada e inclui perguntas diretas e indiretas. As perguntas diretas são aquelas em que o elicitador tem uma idéia clara de qual conhecimento é necessário para completar a base de conhecimentos, com respostas explícitas. As perguntas indiretas são aquelas em que o elicitador deixa o especialista livre para sugerir novos tipos de conhecimento e é aconselhável quando o elicitador não tem idéia precisa do domínio.

- Análise de discurso: Consiste essencialmente em gravar-se a entrevista com o especialista para depois analisar-se a conversação.

- Discussão focalizada: É semelhante a uma entrevista em que existe um ponto, o foco em torno do qual são feitas as perguntas. Normalmente é utilizada em uma fase avançada da elicitação do conhecimento e serve para esclarecer um ponto de dúvida.

- Análise de protocolo: O elicitador cria um problema específico e pede ao especialista para refletir sobre a situação e resolver o problema.

- Ordenamento de cartões: Pode-se escrever os elementos em pequenas fichas que são submetidas ao especialista para classificação. Espera-se que ele possa agrupar os cartões em pilhas, nomear cada pilha e justificar seu agrupamento.

- Geração de matriz: Muitos especialistas costumam colocar suas informações em tabelas. Neste caso, o exame destas tabelas pode ajudar muito a elicitação.

- *Teachback*: É a inversão de papéis. Nela, o elicitador “ensina” ao especialista algum aspecto do conhecimento previamente escolhido e tenta sintonizar o processo com o especialista. Deve ser usada nas fases finais da elicitação.

## 2.5.2 Métodos de Representação do Conhecimento

A parte mais importante no projeto de um SE é a escolha do método de representação de conhecimento. A linguagem associada ao método escolhido deve ser suficientemente expressiva para permitir a representação do conhecimento a respeito do domínio escolhido, de maneira completa e eficiente. Em tese, uma representação geral, como a lógica, seria suficientemente expressiva para representar qualquer tipo de conhecimento. No entanto, problemas de eficiência, facilidade de uso e a necessidade de expressar conhecimento incerto e incompleto levaram ao desenvolvimento de diversos tipos de formalismos de representação de conhecimento (BITTENCOURT, 2001).

As principais formas de representação são:

- Regras de produção;
- Redes semânticas;
- Quadros/*Frames* e Roteiros;
- Lógica das preposições e dos predicados.

Todas as representações são potencialmente importantes, porém, será dada mais ênfase às regras de produção, técnica utilizada no desenvolvimento deste projeto.

As regras de produção foram concebidas pelo matemático Emil Post em 1943, como modelo “computacional” geral de solução de problemas, quando demonstrou que um procedimento calculável pode ser modelado como um sistema de produção. Nas décadas de cinquenta e sessenta, serviram de suporte para representar a forma humana de resolver

problemas, especialmente no jogo de xadrez e na criptoaritmética. Somente na década de setenta é que foram aproveitadas como suporte para modelo mental (RABUSKE, 1995).

O termo “Sistemas de Produção” (SP) é atualmente usado para descrever uma família de sistemas, que têm em comum o fato de serem constituídas de um conjunto de regras, que reúnem condições e ações. A condição é constituída por um padrão que determina a aplicabilidade da regra, enquanto a ação indica o que será realizado quando a regra for aplicada.

Um sistema de produção poderá ser formado por uma ou mais bases de regras, separadas segundo as conveniências de processamento. Complementa, ainda, o sistema de produção, uma estratégia de controle a qual estabelece as prioridades em que regras serão aplicadas, bem como critérios de desempate quando houver mais regras candidatas à aplicação a um só tempo. Este último aspecto denomina-se “resolução de conflitos”.

Os sistemas de produção têm vantagens e desvantagens gerais, que cumpre analisar (RABUSKE, 1995).

Dentre as vantagens destacam-se:

- Modularidade: As regras dos SP podem ser consideradas, para efeito de manipulação, como peças independentes. Como os programas de IA quase sempre estão incompletos, novas regras podem ser acrescentadas ao conjunto já existente, sem maiores preocupações. Esta é uma característica extremamente importante, pois o volume do conhecimento em geral é grande e a verificação de sua consistência, difícil. Para sistemas muito grandes a modularidade é um elemento que aumenta indiretamente a complexidade.

- Naturalidade: Considera-se a regra uma forma natural de pensar a solução de problemas. Por isso, o que deve ser feito em determinada circunstância é diretamente traduzido em regras de produção.

- Uniformidade: Se for observado um SP, notar-se-á que todas as regras estão escritas seguindo-se o mesmo padrão. Esta padronização, por vezes, pode gerar alguns

inconvenientes. Contudo, esta forma rígida de representação permite que pessoas não familiarizadas com o sistema também possam analisar seu conhecimento.

Como desvantagens, tem-se:

- Incompletude: Característica dos SP resultante da modularidade e da uniformidade, fazendo com que seja difícil verificar quão completos estes sistemas são, bem como verificar os possíveis fluxos de processamento. Consegue-se contornar um pouco este problema, se as regras forem separadas em subconjuntos, com o objetivo de clarear seu entendimento.

- Ineficiência: Também resultante da modularidade e da uniformidade. A ineficiência resulta, particularmente, do número de regras a combinar, e também do esforço de *matching* necessário ao suporte de execução das regras. *Matching* entende-se aqui como a verificação das regras que se aplicam ao estado do problema, bem como, a verificação de quais regras antecedem ou sucedem outra regra. Uma forma de diminuir a ineficiência é investir na ordenação apropriada das regras e formas de seleccioná-las.

## **2.6 Tratamento de Incerteza**

A dificuldade, ou a impossibilidade de se obter todas informações e de equacionar a realidade imprecisa do mundo, levou alguns cientistas a propor lógicas alternativas que seriam mais propícias à representação daquele mundo particular (DE AZEVEDO *et al.*, 2000).

Uma característica comum nos SEs atuais é a existência de um mecanismo de raciocínio incerto que permita representar a incerteza a respeito do conhecimento do domínio. Entre esses mecanismos, podemos citar:

- Lógica Nebulosa ou *Fuzzy*;

- Raciocínio Probabilístico e
- Fatores de Certeza.

### 2.6.1 Lógica *Fuzzy*

A Lógica *Fuzzy* foi introduzida por Lofti Zadeh, em 1965 e tem por objetivo permitir graduações na pertinência de um elemento a uma dada classe, ou seja, de possibilitar a um elemento de pertencer com maior ou menor intensidade àquela classe. Basicamente, isso se faz quando o grau de pertinência de um elemento ao conjunto, que na teoria dos conjuntos “clássica” assume apenas os valores 0 ou 1, passa a ser dado por um intervalo de números reais [0,1] (BITTENCOURT, 2001).

Assim, a imprecisão a respeito de uma afirmação é expressa através de um número, que em vez de probabilidade exprime a possibilidade da afirmação ser correta (DE AZEVEDO *et al.*, 2000).

### 2.6.2 Raciocínio Probabilístico

O Raciocínio Probabilístico é talvez o mais antigo método de tratamento de incerteza. É o raciocínio que, se apoiando em informações probabilísticas sobre fatos de um domínio, chega a uma conclusão a respeito de um novo fato, conclusão esta associada a uma probabilidade (BARRETO, 2001).

Probabilidade é a chance que um determinado evento tem de ocorrer, e pode assumir qualquer valor real entre [0,1]. Formalmente, a probabilidade de que um evento X ocorra é dada pela Equação 2.1.

$$P(X) = \frac{\text{Número de resultados possíveis em que X ocorre}}{\text{Número total de resultados possíveis}} \quad \text{Equação 2.1}$$



### 2.6.3 Fatores de Certeza

Em 1976 surgiu, durante o desenvolvimento do SE conhecido como MYCIN, o modelo de tratamento de incerteza conhecido por Fatores de Certeza.

Formalmente, o fator de certeza (FC) é definido em função da medida de crença (MC) e em função da medida de descrença (MD), em uma hipótese H, para uma evidência E, sendo representado pela Equação 2.2.

$$FC(H, E) = MC(H, E) - MD(H, E) \quad \text{Equação 2.2}$$

Onde,

- FC( H , E ) é o fator de certeza na hipótese H para uma mesma evidência E;
- MC( H , E ) é a medida de crença em H, dado E. Este termo é uma medida de até que ponto a evidência sustenta a hipótese;
- MD( H , E ) é a medida de descrença em H, dado E. Este termo é uma medida de até que ponto a evidência sustenta a negação da hipótese.

As medidas de crença e descrença são, por sua vez, dadas pela Equação 2.3 e Equação 2.4 (RICH, 1988).

$$MC(H, E) = \left\{ \begin{array}{l} 1, P(H) = 1 \\ \frac{\max[P(H/E), P(H)] - P(H)}{\max[1, 0] - P(H)}, P(H) \neq 1 \end{array} \right\} \quad \text{Equação 2.3}$$

$$MD(H, E) = \left\{ \begin{array}{l} 1, P(H) = 0 \\ \frac{\min[P(H/E), P(H)] - P(H)}{\min[1, 0] - P(H)}, P(H) \neq 0 \end{array} \right\} \quad \text{Equação 2.4}$$

Onde,

- P(H) é a probabilidade de ocorrência da hipótese H;
- P(H/E) é a probabilidade condicional da ocorrência da hipótese H, dada a ocorrência da evidência E.

## 2.7 Shell

Inicialmente, cada SE era projetado e desenvolvido desde o início. Com o tempo, observou-se que eles apresentavam várias características em comum. Com o objetivo de se aproveitar estas características comuns, simplificando as etapas de desenvolvimento de um SE completo e visando uma maior viabilidade econômica na implementação de um SE, surgiram as *shells* para SE (HAHNE, 2001).

A idéia principal das *shells* é separar a base de conhecimento (parte do sistema que trata especificamente do problema no domínio considerado) da máquina de inferência (parte que move o sistema). O usuário deve se preocupar apenas em obter o conhecimento do especialista humano, pois a máquina de inferência é inerente ao sistema.

Embora deva haver um banco de conhecimentos para cada aplicação, as *shells* são deficitárias a este respeito. Não contêm nenhum conhecimento dependente do domínio, mas sustentam todas as outras facilidades, tais como (CHORAFRAS, 1988):

- Mecanismos de inferência;
- Acesso à base de dados;
- Interface de diálogos de linguagem natural;
- Interfaces procedimentais;
- Facilidade de explicação.

Como vantagens em usar as *shells* pode-se citar (HAHNE, 2001):

- Prototipagem rápida;
- Usam estruturas de dados e conhecimentos pré-definidos;

- Menor necessidade de treinamento de desenvolvedores de SE (é mais simples construir um SE usando uma *shell*).

Existem várias *shells* disponíveis no mercado, como por exemplo: NETICA, SPIRIT, Expert Sinta, etc.

## **2.8 Shell para Sistemas Especialistas Fuzzy**

A *Shell* para Sistemas Especialistas *Fuzzy* (SSEF) foi desenvolvida no Instituto de Engenharia Biomédica (IEB-UFSC), do Departamento de Engenharia Elétrica, da Universidade Federal de Santa Catarina, Brasil (HAHNE, 2001).

As características do sistema desenvolvido foram:

- Tratamento de incerteza utilizando Lógica *Fuzzy*;
- Baseado em regras de produção;
- Para implementar a máquina de inferência foi utilizada a ferramenta Visual Prolog 5.2 Personal Edition;
- A interface foi implementada na ferramenta Borland Delphi 3.0.

O SSEF possibilita criar, modificar, consultar e excluir Sistemas Especialistas.

### **2.8.1 Módulo Criar**

Módulo que é utilizado para criar um novo Sistema Especialista, através da adição de regras e perguntas. A partir desse módulo, é construída a base de conhecimento. No módulo de criação é possível: criar as perguntas com as respectivas explicações e criar as regras. Para as perguntas, é possível definir as opções de resposta e qual o valor de verdade

para cada resposta, além de cadastrar explicações para cada pergunta feita. O SE criado é salvo em um arquivo tipo .ESP referente à especialista. É um arquivo texto com as várias informações fornecidas pelo usuário.

A Figura 2.5 mostra a tela após a entrada dos dados.

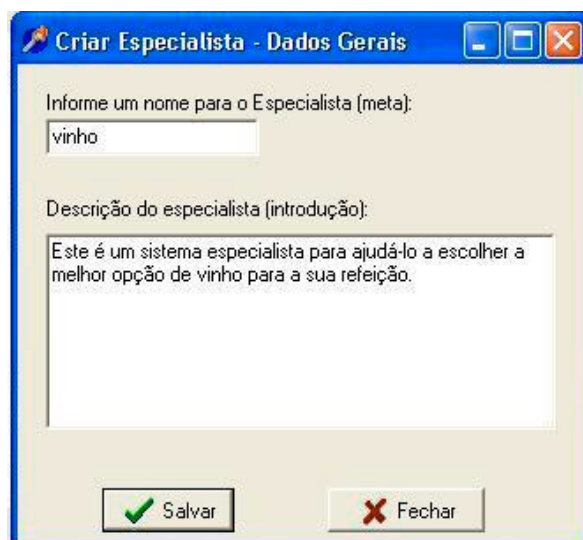


Figura 2.5 – Tela para criação do sistema.

## 2.8.2 Módulo Modificar

Permite modificar as regras ou perguntas de uma base de conhecimento. Através deste módulo pode-se modificar perguntas e regras já existentes ou adicionar novas regras a um SE já existente.

A Figura 2.6 apresenta a tela para gerenciamento das questões.

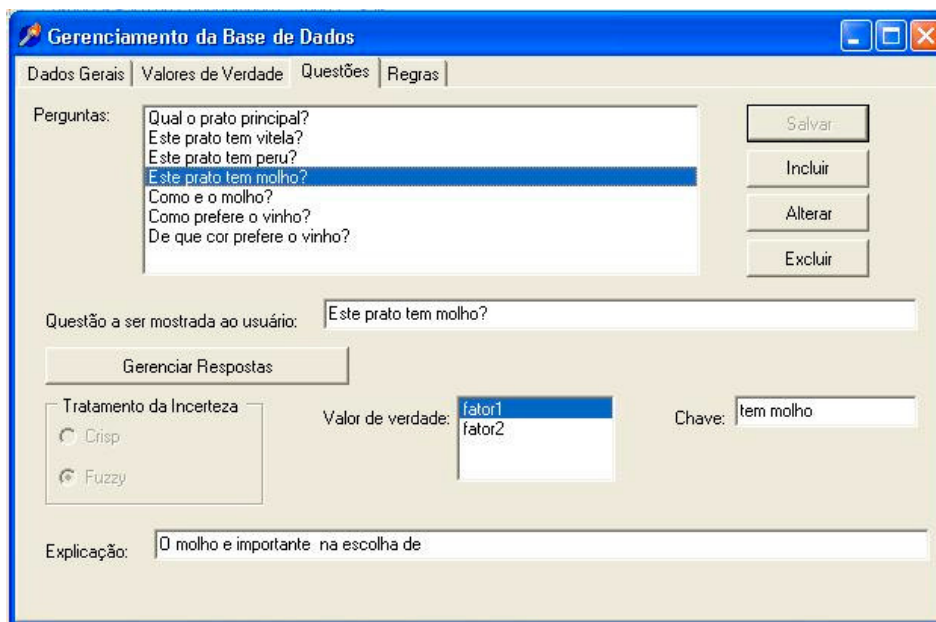
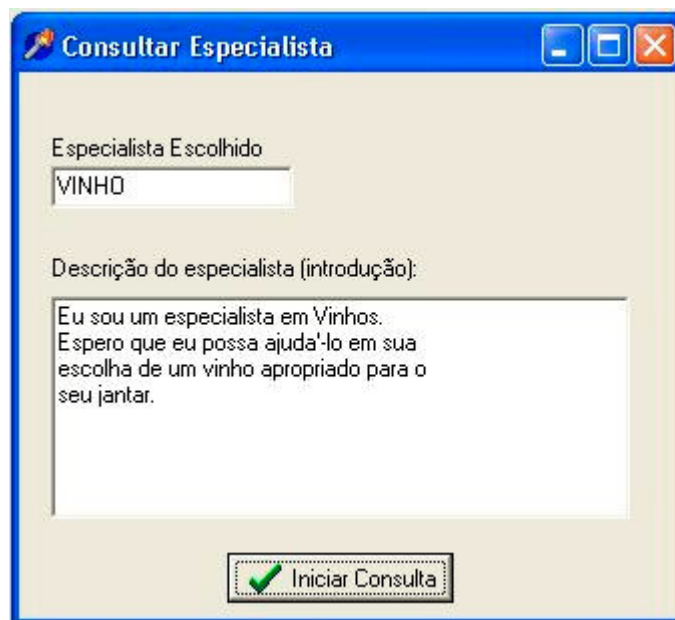


Figura 2.6 – Tela para gerenciamento das questões

### 2.8.3 Módulo Consultar

Permite consultar uma base de conhecimento existente. Neste módulo, o usuário escolhe o especialista que quer consultar. Após realizar esta tarefa, é gerada, automaticamente, uma seqüência de menus de acordo com as perguntas que foram definidas para este especialista. Após responder todas as questões, o sistema informa ao usuário as respostas que se encaixam no quadro apontado.

Após selecionar o arquivo que se deseja consultar, é mostrada ao usuário a tela apresentada na Figura 2.7.

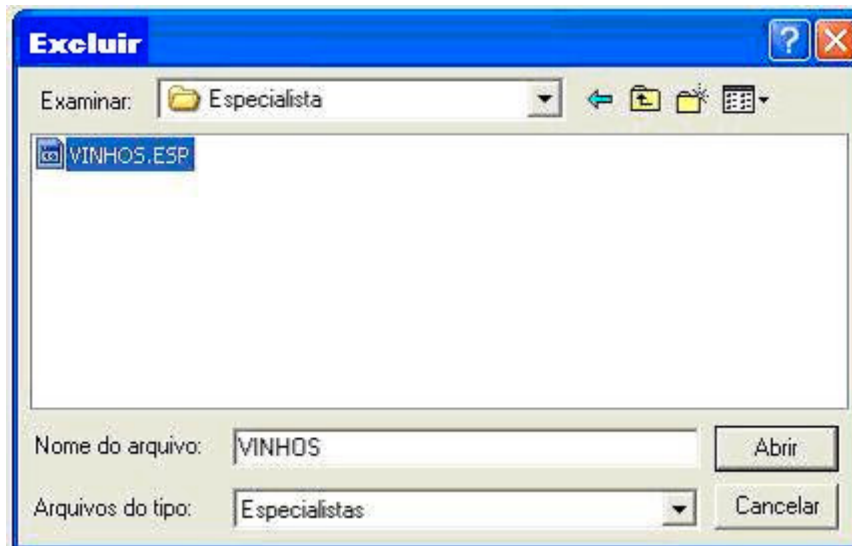


**Figura 2.7 – Tela de consulta do sistema.**

O módulo consultar é o único escrito em Prolog, pois é este o módulo que utiliza a máquina de inferência. Todos os outros módulos são implementados utilizando-se a linguagem Delphi.

#### **2.8.4 Módulo Excluir**

Elimina uma base de conhecimento (ou seja, um “especialista”) criada. A Figura 2.8 mostra a tela para a exclusão de um especialista.



**Figura 2.8 – Tela para exclusão de um especialista.**

Entretanto, a *shell* desenvolvida por Hahne (2001), não apresentava uma interface funcional e, também, não possibilitava a criação de sistemas especialistas com mais de uma meta. Foram esses os motivos que levaram ao desenvolvimento de uma nova *shell*.

### 3 Implementação do Sistema

Neste trabalho foram desenvolvidos dois sistemas, em virtude dos diferentes tipos de usuários. O usuário especialista quer uma ferramenta que o auxilie na construção de um sistema especialista, no caso uma *shell*. E tem o usuário que deseja apenas consultar uma base de conhecimento e armazenar os resultados das consultas, sem a necessidade de saber como o sistema especialista foi criado.

O primeiro sistema tem por objetivo permitir a implementação de um SE (na realidade, a base de conhecimento) e sua posterior consulta, através da geração automática de menus. Esse sistema recebeu o nome de Intellec.

O segundo sistema desenvolvido tem como objetivo cadastrar pacientes e manter um histórico de consultas. Essas características são intrínsecas a sistemas especialistas da área médica, em que o médico tem a necessidade de acompanhar o quadro clínico do paciente ao longo do tempo. Esse sistema recebeu o nome de DBIntellec.

#### 3.1 Intellec

O Intellec está dividido em três módulos, que implementam as funções necessárias para que o sistema execute as tarefas propostas:

- Módulo Editar Base de Conhecimento;
- Módulo Consultar;
- Módulo Depurar.

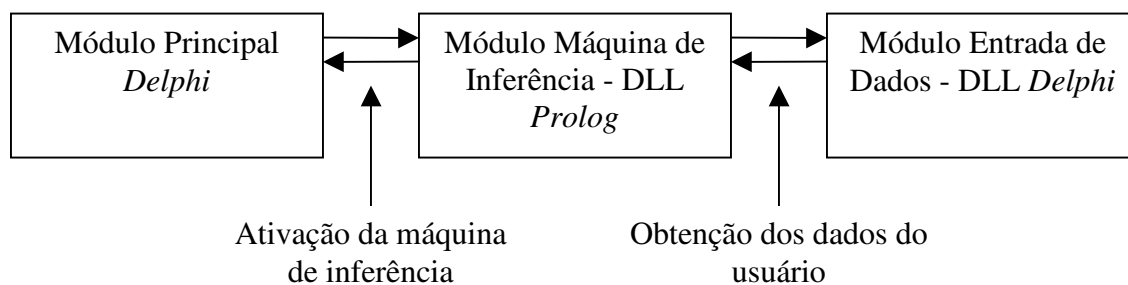
Para a implementação foram utilizadas as ferramentas de programação Visual Prolog 5.2 e Borland Delphi 6.0. A ferramenta Borland Delphi 6.0 é utilizada para implementar os módulos Editar Base de Conhecimento e Depurar. Esta ferramenta foi



escolhida por apresentar inúmeros componentes que facilitam a implementação da interface.

A ferramenta Visual Prolog 5.2 Personal Edition foi utilizada para implementar a máquina de inferência. A sua escolha foi feita por esta ferramenta facilitar o uso da linguagem Prolog, também fornecendo componentes visuais para o desenvolvimento de projetos e DLLs (*Dynamic Link Library*).

O sistema é composto de um arquivo principal executável e duas DLLs. A criação da DLL surgiu da necessidade de se comunicar linguagens diferentes. O programa principal, escrito em Delphi, quando realiza uma consulta a um sistema especialista, chama uma função escrita em Prolog, que será responsável pela ativação da máquina de inferência. Já a máquina de inferência chama funções em Delphi, para obter as respostas necessárias dos usuários. A DLL escrita em Delphi devolve o controle para a DLL escrita em Prolog que, finalmente, devolve o controle para o módulo principal. O esquema pode ser visto na Figura 3.1.



**Figura 3.1 – Esquema de funcionamento do Intellec.**

A partir do módulo principal é possível acionar todos os módulos do sistema, como é apresentado na Figura 3.2.

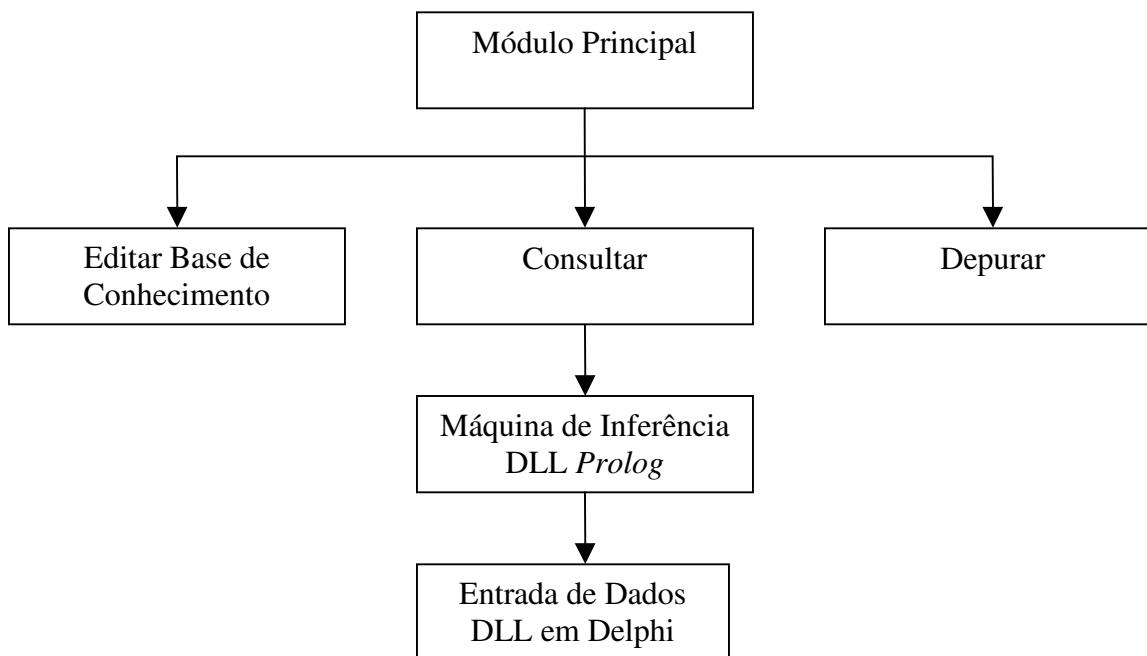


Figura 3.2 – Relacionamento entre os módulos e DLLs do sistema.

A seguir, será descrito cada módulo do sistema e a implementação das DLLs.

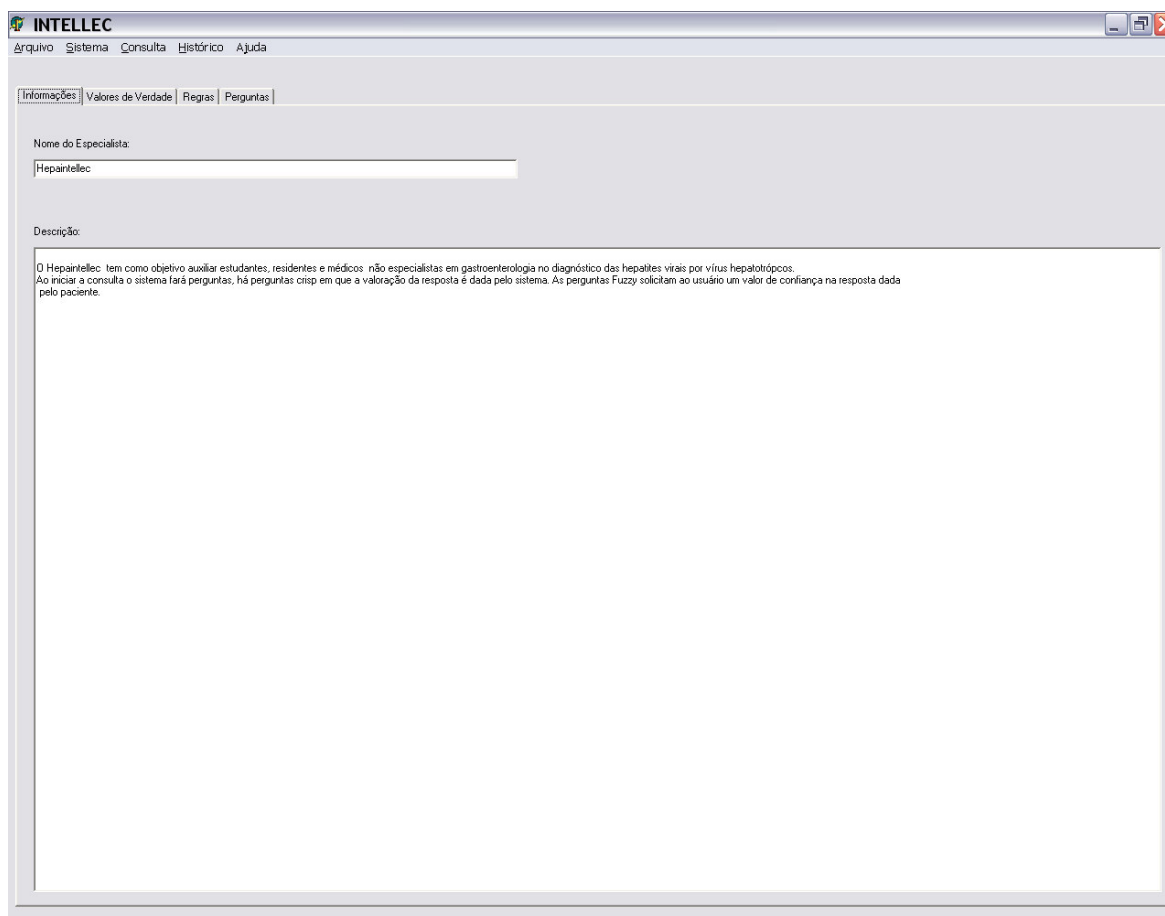
### 3.1.1 Módulo Editar Base de Conhecimento

O módulo Editar Base de Conhecimento tem como objetivo criar e/ou modificar a base de conhecimento de um sistema especialista.

A criação de uma base de conhecimento deve seguir a seguinte seqüência: descrição do sistema especialista que está sendo implementado, definição das variáveis e dos objetivos, elaboração das regras, definição dos valores de verdade e elaboração das perguntas.

### ➤ Descrição do Sistema Especialista

A Figura 3.3 apresenta a tela para o especialista definir qual é o nome da base de conhecimento e também para descrever as informações gerais do sistema especialista que está sendo criado.



**Figura 3.3 – Descrição do Sistema Especialista.**

### ➤ Definição das Variáveis

Antes de criar as regras da base de conhecimento, é necessário que todas as variáveis utilizadas, bem como suas respectivas listas de valores, sejam criadas. A tela de edição de variáveis, apresentada na Figura 3.4, permite ao especialista criar, abrir ou excluir uma variável.

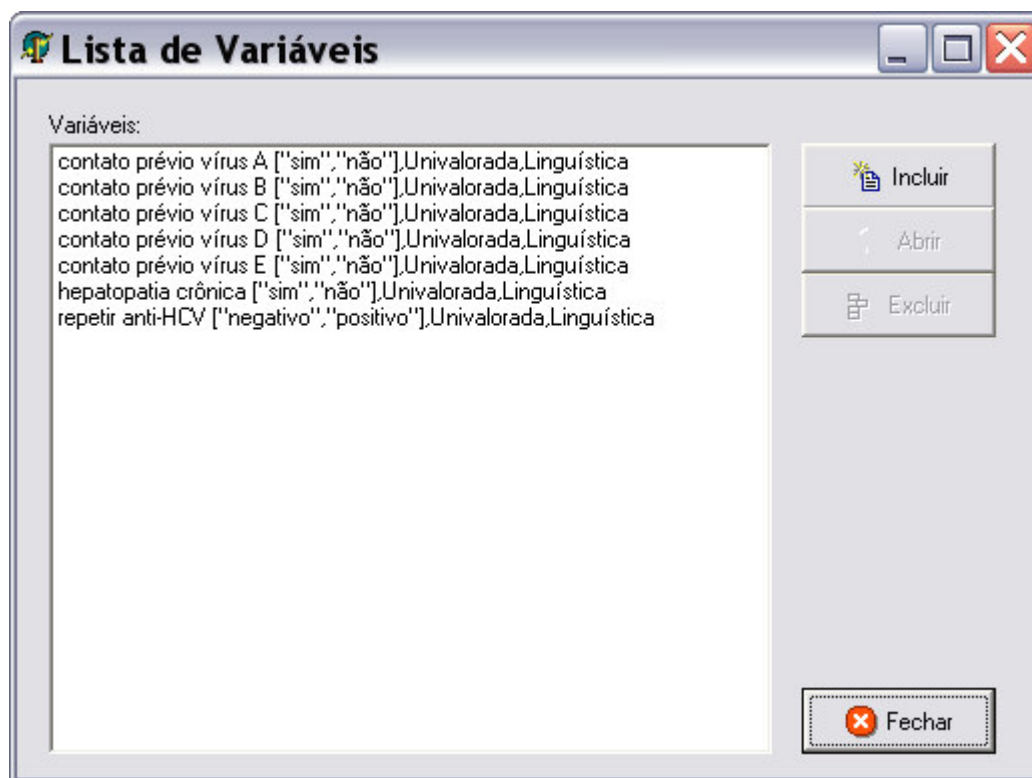


Figura 3.4 – Tela de edição de variáveis.

Para criar uma variável, é necessário definir se ela é uma variável univalorada ou multivalorada. Se a variável for univalorada o sistema aceita apenas uma instanciação por consulta, já a variável multivalorada aceita uma ou mais.

O Intellec ainda permite ao especialista escolher se a variável é do tipo numérica, lingüística ou imagem. Se a variável é do tipo numérica, o especialista não define a lista de valores. O sistema permite ao usuário entrar com valores em tempo de execução. Para a variável do tipo lingüística ou imagem, o especialista deverá definir a lista de valores. No caso da variável do tipo imagem, os valores são os arquivos que contenham as imagens. Estes arquivos podem ter extensão JPEG ou BMP.

A variável imagem é indicada a sistemas especialistas da área médica, principalmente na área de dermatologia, em que uma imagem pode ser muito mais significativa que uma variável lingüística. Isto pode ser verificado através da Figura 3.5, que apresenta uma pergunta feita ao usuário durante a execução de uma consulta, utilizando uma variável do tipo imagem.



Figura 3.5 – Pergunta utilizando variável do tipo imagem. As imagens utilizadas foram retiradas do endereço [http://www.turmadapele.com.br/dicas/fotoprotecao\\_cancer.asp](http://www.turmadapele.com.br/dicas/fotoprotecao_cancer.asp), acessado no dia 10/12/2004.

A Figura 3.6 apresenta a tela para a definição das variáveis.



Figura 3.6 – Variável.

➤ Definição dos Objetivos

Após ter definido as variáveis da base de conhecimento, é necessário definir quais delas serão variáveis metas ou objetivos. A definição dos objetivos irá controlar o modo como a máquina de inferência se comportará.

A Figura 3.7 apresenta a tela para a escolha das variáveis metas.

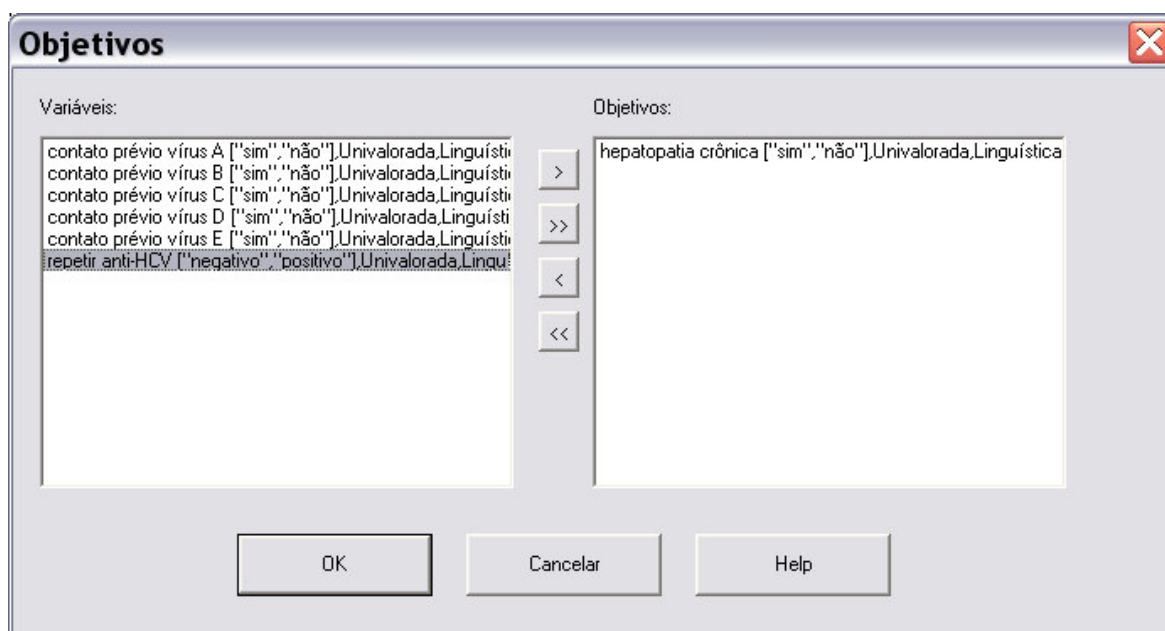


Figura 3.7 – Objetivos.

➤ Elaboração das Regras

Após a definição das variáveis e dos objetivos do sistema especialista, é feita a elaboração das regras. A tela de edição de regras, apresentada na Figura 3.8, permite ao especialista criar, abrir ou excluir uma regra.

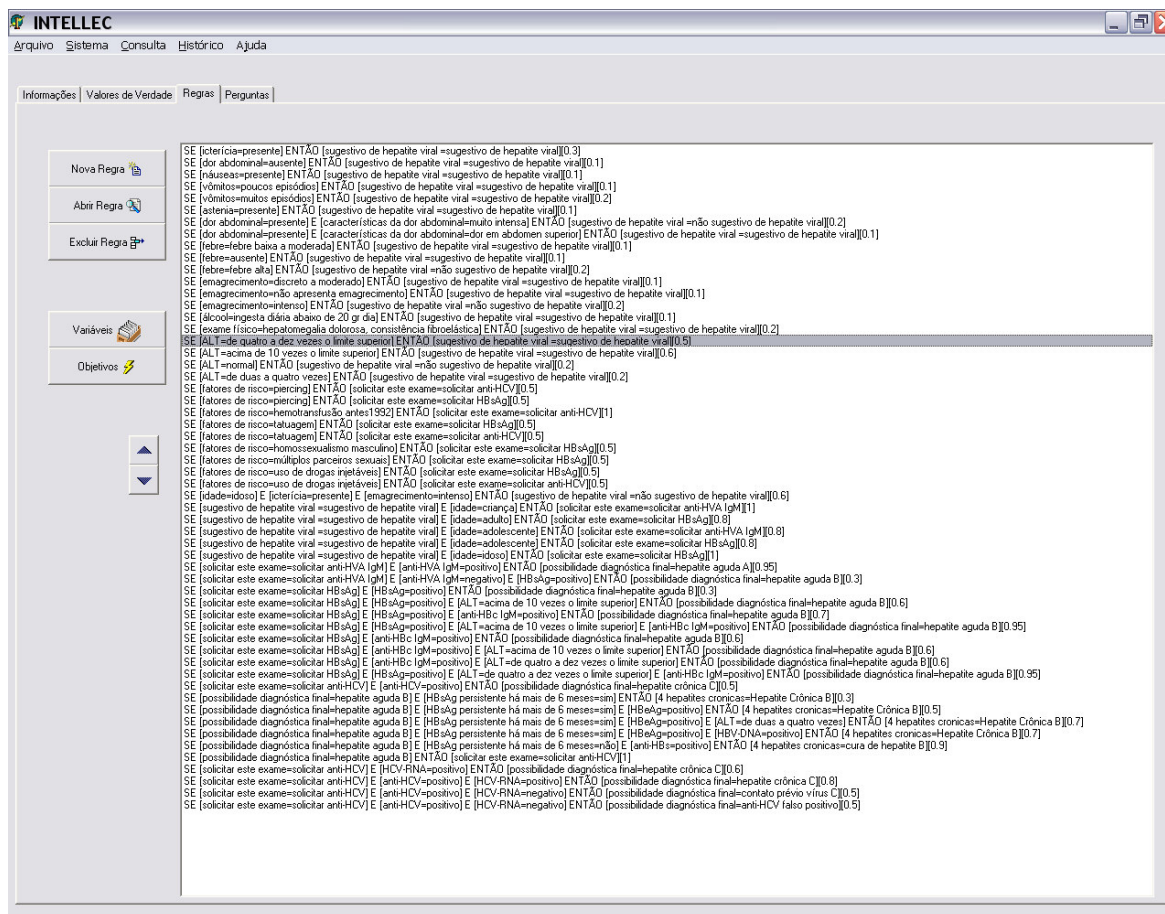


Figura 3.8 – Tela de edição de regras.

As regras têm o formato SE... ENTÃO. Para cada regra, o especialista deverá fornecer a lista de premissas, a conclusão e o valor de verdade. A Figura 3.9 apresenta a tela de criação de regras.

**Regra**

Premissas:

Variável: Valor da variável:

dor abdominal=presente  
características da dor abdominal=muito intensa

Conclusões:

Variável: Valor da variável:

sugestivo de hepatite viral =não sugestivo de hepatite viral

Valor Verdade: 0.2

Cancelar OK

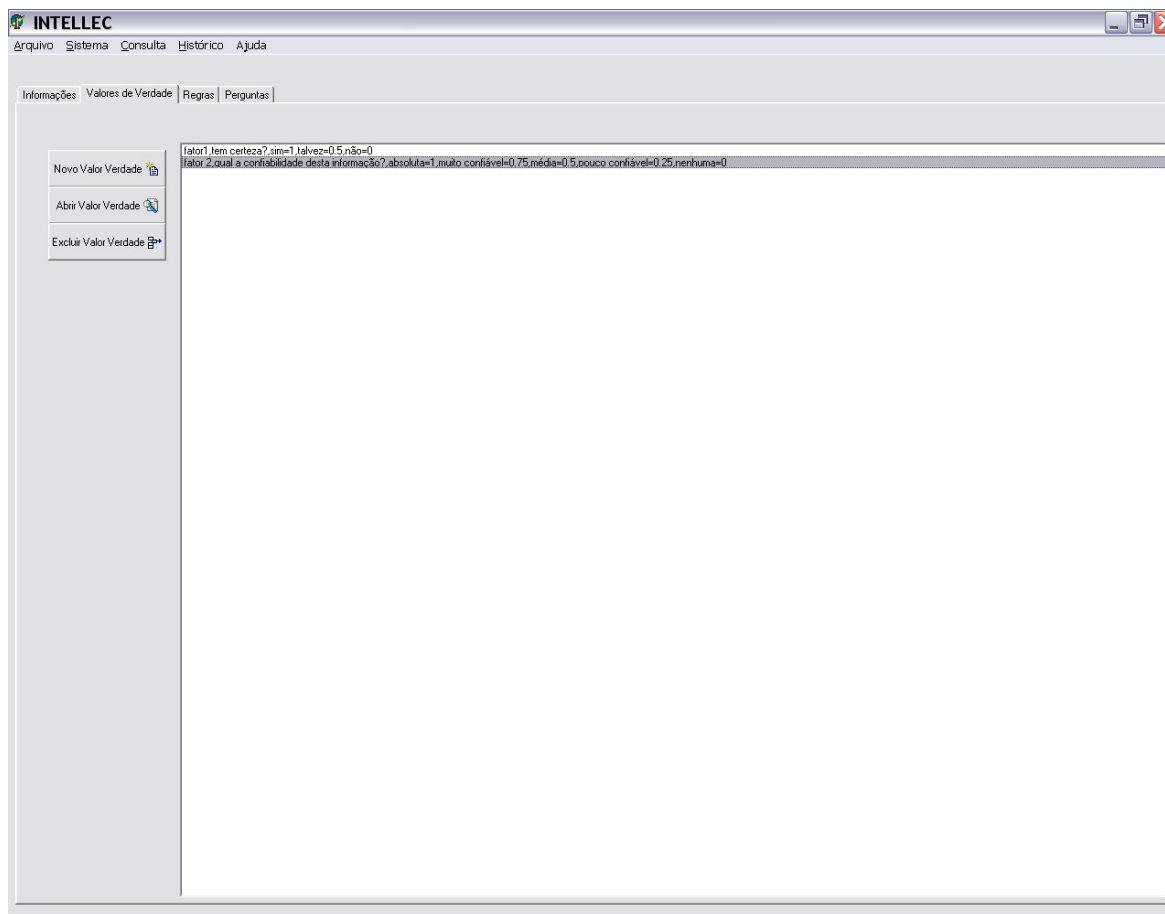
**Figura 3.9 – Regras.**

➤ Definição dos Valores de Verdade

A definição dos valores de verdade permite ao usuário associar graus de confiança a cada uma das respostas assinaladas. Esses valores de verdade são utilizados para as perguntas com tratamento de incerteza *fuzzy*.

A Figura 3.10 apresenta a tela de edição de valores de verdade. Nesta tela o especialista pode criar, abrir ou excluir um valor de verdade.





**Figura 3.10 – Tela de edição de valores de verdade.**

Para criar um novo valor de verdade, o especialista deverá fornecer o nome, a pergunta e a lista de termos do valor de verdade. A Figura 3.11 apresenta um exemplo de valor de verdade, em que o especialista atribuiu ao campo *Nome* o valor “fator2”, ao campo *Pergunta*, “Tem certeza” e a *Lista de Termos*, “absoluta = 1, muito confiável = 0,75, média = 0,5, pouco confiável = 0,25 e nenhuma = 0”.

**Valor Verdade**

Nome: fator 2

Pergunta: qual a confiabilidade desta informação?

Termo:

- absoluta=1
- média=0.5
- muito confiável=0.75**
- nenhuma=0
- pouco confiável=0.25

Incluir

Excluir

Alterar

OK

Cancelar

Termo relativo ao valor verdade: muito confiável

Valor do fator de certeza (entre 0 e 1): 0.75

**Figura 3.11 – Valor verdade.**

➤ **Elaboração das Perguntas**

A elaboração das perguntas é feita através da tela de edição de perguntas, apresentada na Figura 3.12. Esta tela permite ao especialista criar, abrir ou excluir uma pergunta.

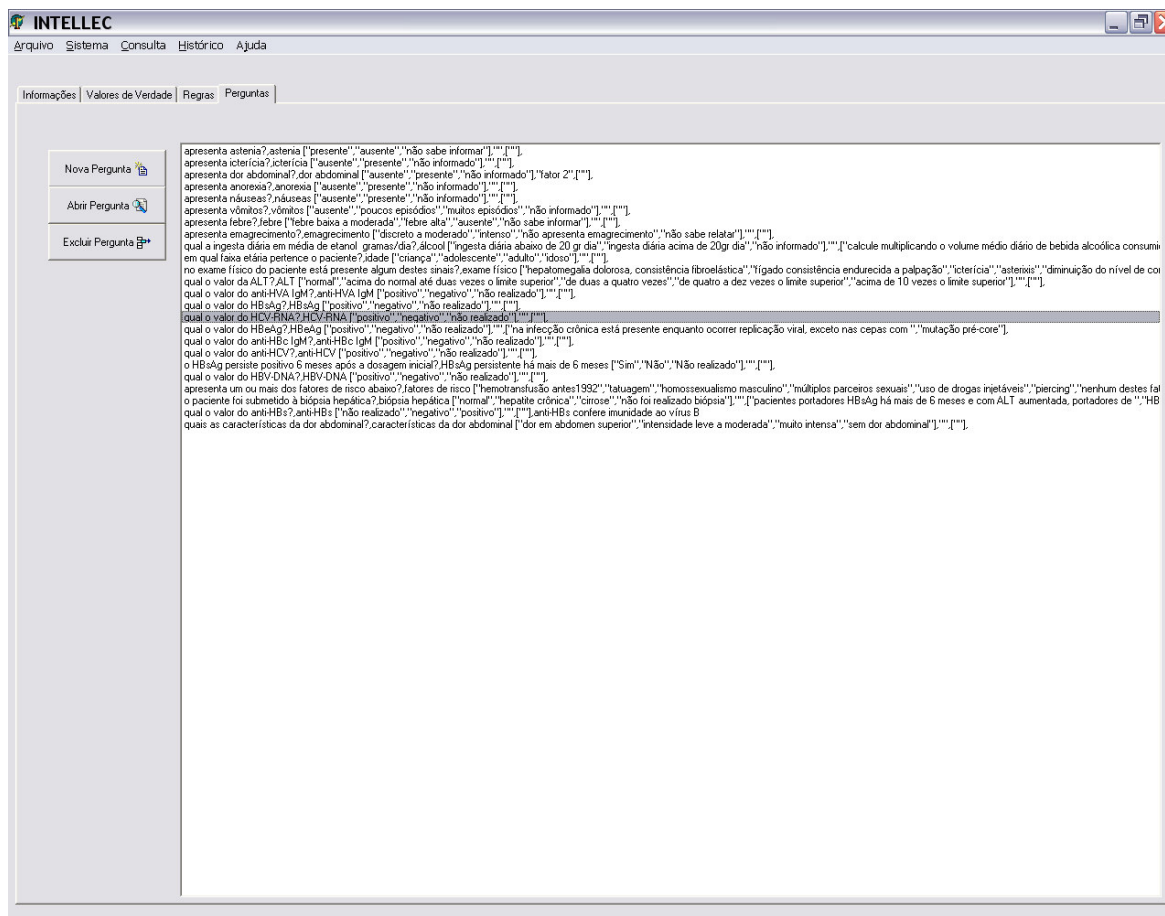


Figura 3.32 – Tela de edição de perguntas.

Cada pergunta está associada a uma variável do sistema especialista que será feita durante a execução de uma consulta. Para cada pergunta criada, o especialista deverá fornecer a pergunta que será mostrada ao usuário, o tipo de tratamento de incerteza (*Crisp* ou *fuzzy*), informar uma explicação que será utilizada para explicar ao usuário o porquê da necessidade daquela pergunta e ainda, um arquivo tutorial.

Para uma pergunta *crisp* o valor de verdade é 1. Para uma pergunta *fuzzy*, o valor de verdade é aquele fornecido pelo usuário.

A opção de incluir um tutorial para cada pergunta, vem da necessidade dos especialistas em construir sistemas para a área da educação. O arquivo tutorial pode ter extensão HTML (*Hipertext Markup Language*), PDF (*Portable Document Format*) ou arquivos gerados pelo Microsoft Office.

A Figura 3.13 apresenta a tela para a criação de perguntas do sistema especialista.

**Figura 3.13 – Perguntas.**

➤ Arquivo \*.ESP

O arquivo gerado pelo Intellec possui extensão ESP. Este arquivo contém a base de conhecimento do SE e apresenta-se em forma de predicados, pois esta base é consultada através do predicado *consult* (comando da Prolog que lê os fatos de um arquivo texto) e é adicionada à base de conhecimento do sistema. Os seguintes predicados estão presentes no arquivo:

- *introducao* (lista): Este predicado tem as informações gerais sobre o SE que se está consultando. Contém apenas um argumento, que é uma lista de strings.

- nomeesp(string): Este predicado representa o nome do sistema especialista. Contém apenas um argumento, que é uma string.

- pergunta (inteiro, string, lista, string, lista, lista, string): Este predicado representa as perguntas que devem ser feitas ao usuário durante uma consulta. Ele é composto por sete argumentos, na seguinte ordem:

INTEIRO = Número da pergunta, que é apenas um seqüencial referente ao número total de perguntas;

STRING = A pergunta que será mostrada pela interface ao usuário;

LISTA = A lista de respostas possíveis para a pergunta;

STRING = Pergunta referente aos valores de verdade;

LISTA = Lista dos valores de verdade, representadas por nomes;

LISTA = Lista dos valores de verdade, representada por números;

STRING = Variável que representa a pergunta.

No caso da pergunta ser *crisp*, os argumentos que representam a pergunta referente aos valores de verdade e a lista dos valores de verdade representados por nomes e por números são brancos. Estes argumentos são utilizados somente para perguntas *fuzzy*.

- regra (inteiro, string, string, string, real, lista): Este predicado representa as regras que serão utilizadas para dar uma resposta ao usuário após uma consulta. Ele é composto por seis argumentos, na seguinte ordem:

INTEIRO = Número da regra, que é apenas um seqüencial referente ao número total de regras;

STRING = Representa o operador da comparação, isto é, estabelece uma relação entre a variável e o valor da variável da conclusão (igualdade, diferença, etc.);

STRING = Variável da conclusão;

STRING = Resposta possível para a variável da conclusão;

REAL = Valor de verdade atribuído à conclusão;

LISTA = Contém as premissas da regra. Cada premissa deve informar o operador da comparação, a variável e o valor da variável.

- explicacao (inteiro, lista): Informa uma explicação associada à pergunta, para informar ao usuário qual o motivo da necessidade daquela resposta. Possui dois argumentos, na seguinte ordem:

INTEIRO = Seqüencial que associa a explicação à pergunta;

LISTA = Lista de linhas da explicação.

- tutorial(inteiro, string): Informa o arquivo tutorial que está associado à pergunta. Possui dois argumentos, na seguinte ordem:

INTEIRO = Seqüencial que associa o arquivo tutorial à pergunta;

STRING = Nome do arquivo.

- variavel(string, lista, string, string): Este predicado representa as variáveis do sistema. Ele é composto por quatro argumentos, na seguinte ordem:

STRING = Nome da variável;

LISTA = Contém os valores da variável;

STRING = Indica se a variável é do tipo univalorada ou multivalorada;

STRING = Indica se a variável é do tipo numérica, lingüística ou imagem.

No caso da variável ser numérica, a lista dos valores da variável é branco. Este argumento só é utilizado para variáveis lingüísticas e imagens.

- meta (string): Este predicado representa a variável objetivo do sistema. Ele é composto de apenas um argumento do tipo string, que é a variável objetivo.

No relatório de pesquisa, RP/M – IEB-UFSC – 01/2004, encontra-se a base de conhecimento montada para teste, com o exemplo dos vinhos, sendo possível observar todos estes predicados.

### **3.1.2 Módulo Consultar**

Este módulo é composto por duas DLLs. Uma é a máquina de inferência, implementada em Visual Prolog, e a outra, implementada em Delphi, é utilizada para buscar as informações necessárias do usuário. Este módulo pode ser acionado pela opção

do menu *Consulta/Iniciar*.

Através deste módulo, é possível consultar uma base de conhecimento existente, escolhendo a base de conhecimento desejada. Após informar este dado, é gerada, automaticamente, uma seqüência de menus de acordo com as perguntas que foram definidas para esta base de conhecimento. Após responder todas as questões, o sistema informa ao usuário, com base nas regras que foram cadastradas, as respostas que se encaixam no quadro apontado.

O esquema básico de funcionamento do módulo Consulta consiste da chamada da DLL Prolog, que é chamada *MaquinaInferencia.dll*. Esta DLL passa então a ter o controle do programa, chamando a DLL implementada em Delphi, que é chamada *ObterRespostas.dll* cada vez que for necessário obter algum dado do usuário.

Quando a opção *Consulta/Iniciar* é acionada, pergunta-se ao usuário qual a base de conhecimento que ele deseja consultar, permitindo que ele selecione um arquivo .ESP.

Após selecionar o arquivo, é mostrado ao usuário a tela apresentada na Figura 3.14. A partir dela, o usuário pode iniciar a consulta, realizando quantas consultas desejar.

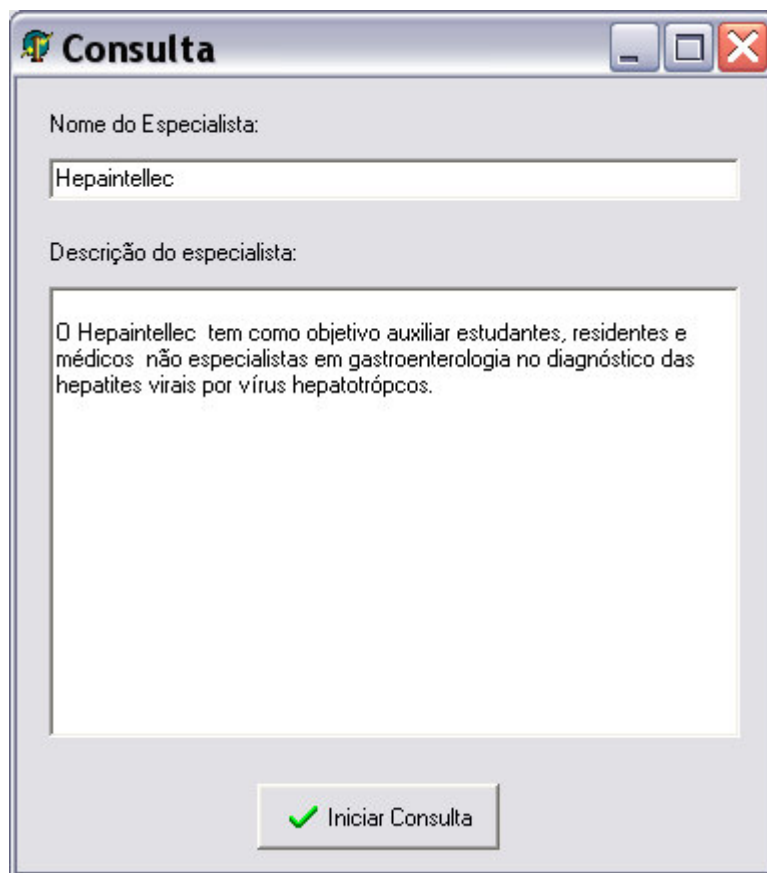


Figura 3.44 – Tela de consulta do sistema.

➤ Módulo Máquina de Inferência - DLL Prolog

A DLL implementada em *Visual Prolog*, chamada *MaquinaInferencia.dll*, é a máquina de inferência do sistema. A função pública na DLL, que é chamada pelo arquivo principal, é *consultar (nomeArquivo, valorRetorno)*, onde o parâmetro *nomeArquivo* é o nome do arquivo que se está consultando e o parâmetro *valorRetorno* é apenas um valor que retorna zero ou um, indicando se houve ou não algum erro durante a execução do programa.

Quando o usuário decide realizar uma consulta, a DLL é acionada pelo executável Delphi. Após carregar a base de dados através do predicado *consult*, utilizando como argumento o nome do arquivo passado como parâmetro, procura-se na base de dados qual é a variável meta do sistema. Por exemplo, *vinho*, no caso de um SE para escolher o melhor vinho para uma refeição. Após verificar qual é a meta, é feita uma procura pela primeira regra que tenha esta variável como conclusão. Ao achar esta regra, ele passa a



tratar com cada premissa individualmente, obtendo-se o valor de verdade para cada uma delas. Então, verifica-se se a variável deve ser perguntada ao usuário, ou seja, se é uma variável de interface, ou se ela deve ser inferida a partir de outras regras.

Considerando-se o primeiro caso, ou seja, uma variável que deve ser perguntada ao usuário, a máquina de inferência verifica se é uma pergunta *crisp* ou *fuzzy*, e chama a DLL Delphi através da função adequada levando como parâmetro o número da pergunta e trazendo de volta a resposta dada e o valor de verdade.

Agora, considerando o segundo caso, ou seja, se a variável deve ser inferida a partir de outras regras, a máquina procura por uma regra que tem esta variável como conclusão. Para esta variável, também é obtido o valor de verdade, sendo que esta pode depender de variáveis que devem ser inferidas a partir das regras existentes, e também de variáveis que devem ser solicitadas ao usuário.

Sabendo todos os valores de verdade para todas as premissas, a máquina de inferência calcula o valor de verdade da regra utilizando a Equação 3.1:

$$\mu(\text{regra}) = \mu(\text{conclusão}) \cdot \min[\mu(\text{premissa1}), \mu(\text{premissa2}), \dots, \mu(\text{premissaN})]$$

Equação 3.1

onde  $\mu$  representa o valor de verdade.

- $\mu(\text{conclusão})$  é o valor de verdade da conclusão fornecido pelo especialista na montagem da base de conhecimento;
- $\mu(\text{premissa1}), \mu(\text{premissa2}), \mu(\text{premissaN})$  são os valores de verdade obtidos pela máquina de inferência para a premissa;
- $N$  é o número de premissas para a regra.

A Equação 3.1 apresenta duas implementações diferentes para o  $e$  ( $\wedge$ ) *fuzzy*. Uma é a combinação das premissas, com um  $\wedge$  definido como o mínimo ( $A \wedge B = \min[A,B]$ ), que combina o valor de verdade das premissas.

Outra é a implementação do  $\wedge$  definido pelo produto ( $A \wedge B = A \cdot B$ ), que combina o valor de verdade da regra com o valor de verdade obtido a partir das premissas

Pela Equação 3.1, observa-se que a verdade da conclusão sempre varia quando as verdades das regras variam (fornecendo o mínimo diferente de zero). Por outro lado, a verdade da composição das premissas nunca é maior que a menor das verdades dos componentes (ALGARVE, 1995, DE AZEVEDO *et al.*, 2004).

O valor de verdade de uma variável que é obtida do usuário é o valor informado pelo próprio usuário. Se a variável é inferida de outras regras, o valor de verdade também é obtido a partir da Equação 3.1.

Quando uma conclusão é pesquisada, todas as regras relacionadas são usadas e a conclusão reflete a interação global entre um conjunto de regras e a informação apresentada pelo usuário. Quando muitas regras indicam a mesma conclusão, a conclusão torna-se mais forte. Além disso, regras mais fortes resultam em conclusões mais fortes. Neste caso, vários determinantes fracos podem ou não vencer um forte. Isto é a força e a atração da Lógica *Fuzzy* (ALGARVE, 1995, DE AZEVEDO *et al.*, 2004).

O programa então salva o resultado após cada conclusão obtida. O processo é repetido para uma nova regra que tenha como conclusão a meta do sistema. A máquina de inferência do *Visual Prolog* então pesquisa outra solução e assim sucessivamente. Quando a *Prolog* testar todos os caminhos possíveis para a variável meta, ou seja, quando se obtém várias regras com a mesma conclusão, os resultados são combinados utilizando um *ou* ( $\vee$ ) *fuzzy*, que combina os valores de verdade da conclusões obtidas duas a duas, através da Equação 3.2:

$$\mu(\text{final}) = \mu(\text{conclusao1}) + \mu(\text{conclusao2}) - \mu(\text{conclusao1}) * \mu(\text{conclusao2})$$

Equação 3.2

onde,

- $\mu(\text{final})$  é o valor de verdade final para uma determinada resposta possível;
- $\mu(\text{conclusão1})$  e  $\mu(\text{conclusão2})$  são valores de verdade intermediários obtidos para aquela resposta.

A Equação 3.2 é obtida a partir da seguinte derivação, considerando-se o  $\vee$  dado pelo produto:

$$\begin{aligned}
 A \vee B &= \neg(\neg(A \vee B)) = \neg(\neg A \wedge \neg B) \\
 \mu(A \vee B) &= \mu[\neg(\neg A \wedge \neg B)] \\
 \mu(A \vee B) &= 1 - \mu(\neg A \wedge \neg B) \\
 \mu(A \vee B) &= 1 - \mu(\neg A) \cdot \mu(\neg B) \\
 \mu(A \vee B) &= 1 - [1 - \mu(A)][1 - \mu(B)] \\
 \mu(A \vee B) &= 1 - 1 + \mu(A) + \mu(B) - \mu(A) \cdot \mu(B)
 \end{aligned}$$

obtendo finalmente:

$$\mu(A \vee B) = \mu(A) + \mu(B) - \mu(A) \cdot \mu(B)$$

Algumas outras observações que devem ser feitas com relação à máquina de inferência:

- Quando uma variável é perguntada ao usuário, o valor recebido como resposta é armazenado na base de dados, de modo que a pergunta não seja feita duas vezes, ou seja, antes de fazer uma pergunta ao usuário, o programa verifica se esta pergunta já havia sido feita, fazendo uma busca na base de dados;

- A máquina salva em um arquivo texto todas as regras que foram aceitas pela *shell* e que foram usadas para se chegar à resposta final;

- Quando a premissa é uma negação, por exemplo, *not corpreferida=tinto*, o valor obtido como valor de verdade do usuário é subtraído de 1. Então, se o usuário responde que *corpreferida=tinto* com 0,7 de valor de verdade, o valor de verdade para a premissa *not corpreferida=tinto* será considerado = 1 - 0,7, ou 0,3.

O arquivo fonte desta DLL se encontra no relatório de pesquisa RP/M – IEB-UFSC – 01/2004.

➤ Módulo Entrada de Dados - DLL Delphi

A DLL implementada em Delphi é acionada pela DLL em *Prolog* sempre que for necessário ao SE consultar o usuário. Ela implementa perguntas *crisp* ou *fuzzy*, dependendo da informação que recebeu da máquina de inferência. A DLL tem duas funções públicas chamadas *perguntaCrisp(numeroPergunta, resposta, valorverdade)* e *perguntaFuzzy(numeroPergunta, resposta, valorverdade)*. Estas duas funções recebem como parâmetro o número da pergunta que será feita para o usuário, e retorna a resposta e o valor de verdade. Para uma pergunta *crisp* o valor de verdade é 1. Para uma pergunta *fuzzy*, o valor de verdade é aquele fornecido pelo usuário.

A Figura 3.15 apresenta a tela de uma pergunta *fuzzy* e a Figura 3.16, de uma pergunta *Crisp*.

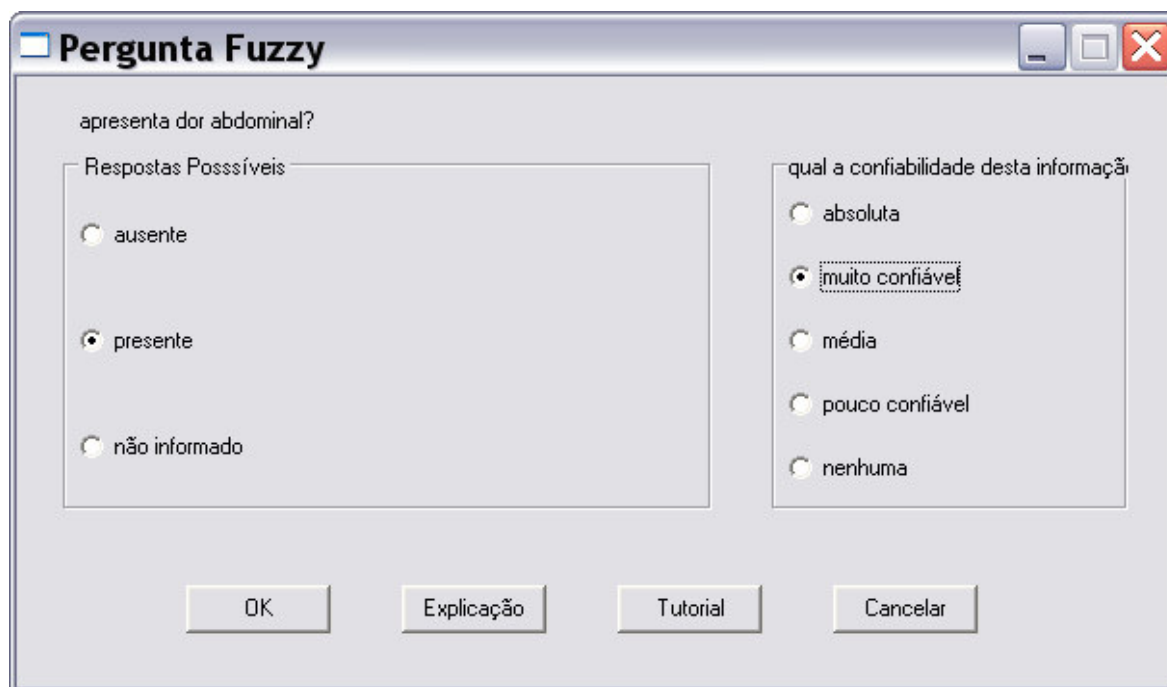


Figura 3.15 – Pergunta *fuzzy*.



**Figura 3.16 – Pergunta *crisp*.**

### 3.1.3 Módulo Depurar

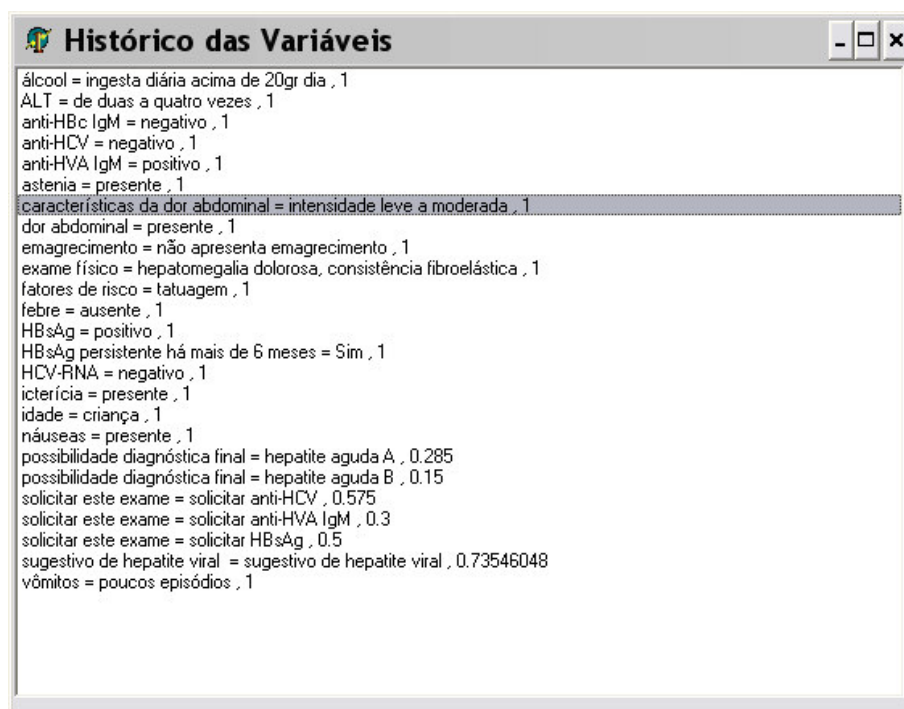
O módulo *Depurar* tem como objetivo apresentar ao usuário, após este realizar uma consulta, quais as regras que foram aceitas para a computação da verdade e também apresentar os valores de verdade para cada variável.

Acionando a opção do menu *Histórico/Regras*, o sistema apresenta a Figura 3.17. Nesta tela são mostradas as regras aceitas pelo sistema após a realização de uma consulta.



**Figura 3.57 – Regras aceitas pelo sistema.**

Acionando a opção do menu *Histórico/Variáveis*, o sistema apresenta a Figura 3.18. Nesta tela são mostrados os valores das variáveis do sistema especialista após a realização de uma consulta.



**Figura 3.18 – Valores das variáveis.**

## **3.2 DBIntellec**

O DBIntellec está dividido em dois módulos, que implementam as funções necessárias para que o sistema seja capaz de realizar as tarefas propostas:

- Módulo Cadastro de Pacientes;
  
- Módulo Histórico de Consultas.

Para a implementação da interface com o usuário foi utilizada a linguagem Borland Delphi 6.0. Foi utilizado o banco de dados Paradox 7 para armazenar os dados dos pacientes e também as consultas realizadas pelo usuário. A escolha pelo banco de dados Paradox 7 se deve ao fato do sistema implementado necessitar armazenar os dados localmente, já que os dados não serão disponibilizados e também por ser uma ferramenta compatível com o Borland Delphi 6.0.

Pelo fato da máquina de inferência ter sido implementada como uma DLL, permitiu que este arquivo fosse utilizado em conjunto com este sistema e também para qualquer outra aplicação. Porém, isso só se torna possível se a base de conhecimento for criada a partir do Intellec.

### **3.2.1 Módulo Cadastro de Pacientes**

O módulo Cadastro de Pacientes tem como objetivo incluir, abrir ou excluir um cadastro de pacientes. A Figura 3.19 apresenta a tela para a edição desse cadastro.

Para a inclusão de um cadastro são obrigatórios o preenchimento dos campos Nome, Idade, Endereço e Cidade, e facultativos, Telefone, Profissão, Sexo e Histórico.

The screenshot shows a software window titled "DBINTELLEC" with a sub-header "Cadastro". There are two tabs: "Informações" (selected) and "Consultas". The form contains the following fields:

- Código do Paciente: 9
- Nome: \* (Text input: Giselle Lopes Gerrari)
- Idade: \* (Text input: 25)
- Endereço: \* (Text input: Jacinto Antunes da Silva, 455)
- Cidade: \* (Text input: Curitiba)
- Telefone: (ex: 3333333) (Text input: 3332016)
- Profissão: (Text input: Engenheira Eletricista)
- Sexo: (Dropdown menu: feminino)
- Histórico: (Large empty text area)

At the bottom left, there is a note: "\* Campos obrigatórios" and a button labeled "Alterar Cadastro" with a refresh icon.

Figura 3.69 – Informações sobre o paciente.

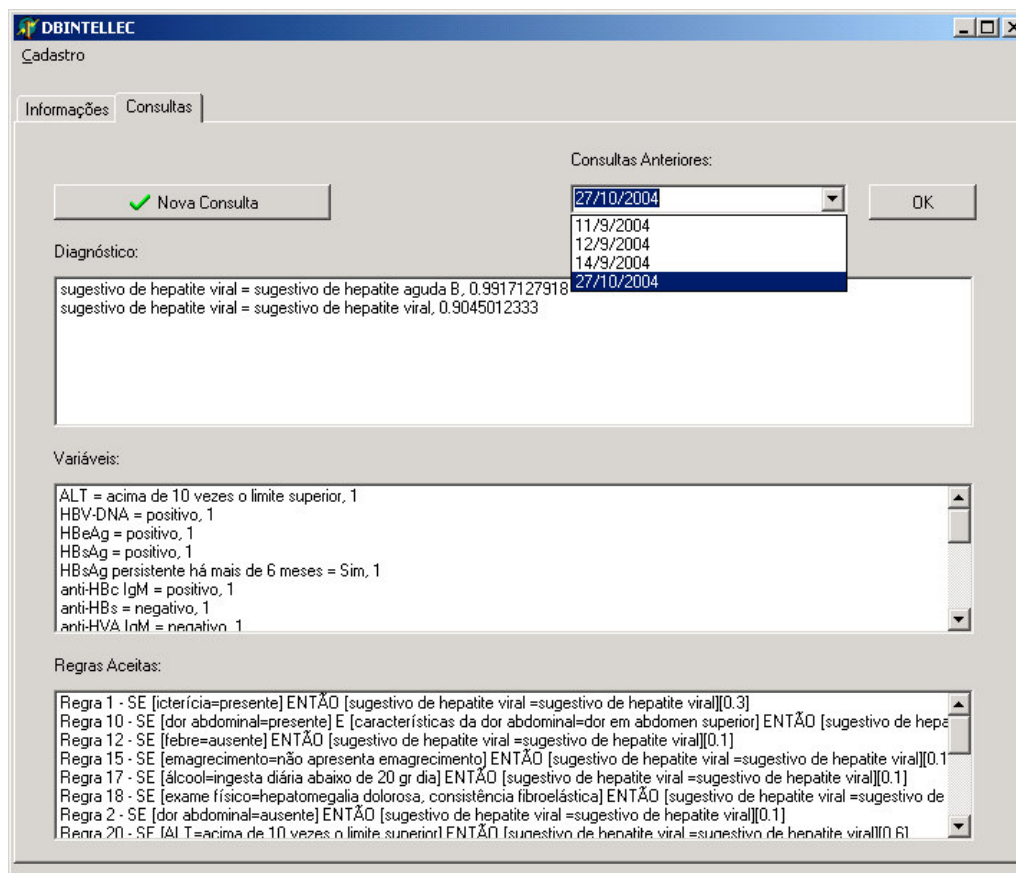
### 3.2.2 Módulo Histórico de Consultas

Após abrir um cadastro de paciente existente, o módulo Histórico de Consultas, apresentado na Figura 3.20, permite ao usuário realizar uma nova consulta ou analisar as consultas realizadas anteriormente através da data que foi realizada a consulta.

Quando o usuário deseja realizar uma nova consulta, o sistema pede que ele informe qual a base de conhecimento que ele deseja consultar, e em seguida é gerada a seqüência de menus de acordo com as perguntas que foram definidas para este especialista. Ao final das perguntas, o sistema apresenta as respostas que se encaixam no quadro apontado e armazena os dados desta consulta no banco de dados.



Para analisar as consultas anteriores, o usuário seleciona a data em que foi realizada a consulta e, o sistema apresenta qual foi o diagnóstico, os valores das variáveis da base de conhecimento e quais foram as regras aceitas para a computação da verdade.



**Figura 3.20 – Histórico das consultas.**

## 4 Resultados e Discussão

Com o objetivo de validar a máquina de inferência, dois problemas foram implementados usando o Intellec e o Expert SINTA, de modo a serem feitos alguns testes e comparações. A escolha pelo Expert SINTA para fazer a validação do sistema, se deve ao fato desta ser uma ferramenta que possui bastante aceitação tanto a nível acadêmico quanto comercial no Brasil e em outros países da América Latina.

O primeiro exemplo, utiliza uma base de conhecimento disponibilizada por ARARIBÓIA (1989), que se constitui na escolha do melhor vinho para uma refeição. O outro exemplo trata de um problema na área médica: auxílio a diagnóstico de hepatites virais. A base de conhecimento deste último exemplo foi implementada por Ramos (2004).

Para ambos os sistemas, procurou-se implementar as mesmas regras e variáveis, sempre procurando corresponder os valores de verdade utilizados pelo Intellec com os fatores de certeza utilizados pelo Expert SINTA. O número de regras variou, devido a limitações de uma ou outra *shell*. A ordem das regras também foi mantida para as duas implementações, por esta ser importante nos dois casos, já que é a ordem das regras que define a ordem na qual as perguntas serão feitas ao usuário, quando este estiver realizando uma consulta.

Para cada problema, foram realizados dois exemplos de consulta, informando-se às duas *shells* a mesma situação. As tabelas comparativas dos resultados obtidos são apresentadas. Deve-se levar em consideração o fato de que os valores de certeza para o Expert SINTA variam de 0 a 100, e para o Intellec, os valores de verdade variam de 0 a 1.

### 4.1 Sistema Especialista em Vinhos

Este é um exemplo de conhecimento empírico. Consiste em se escolher o melhor vinho para uma determinada refeição. O usuário precisa entrar com informações a respeito

do tipo de refeição que será servida e a respeito do seu gosto sobre vinhos. A base de conhecimento deste exemplo foi tirada do livro de ARARIBÓIA (1989). As implementações feita utilizando o Intellec e o Expert SINTA, encontram-se no relatório de pesquisa RP/M – IEB-UFSC – 01/2004.

Neste exemplo foram definidos três variáveis objetivos que são: cor recomendada, doçura recomendada e vinho.

Caso 1: Considera-se uma refeição que tenha como prato principal peixe com molho temperado. A preferência do usuário com relação à cor do vinho é branco, com doçura suave. Neste caso, foi considerado que as respostas dadas pelo usuário foram consideradas com valor de verdade igual a 1 para o tipo de comida e o tipo de molho e com 0,9 de valor de verdade para a doçura e cor preferidas e para a escolha de ter molho na refeição. No Expert SINTA, foram utilizados os valores para fatores de certeza 100 e 90.

Os resultados obtidos estão apresentados na Tabela 4.1, Tabela 4.2 e Tabela 4.3.

**Tabela 4.1 – Resultado obtido para a primeira variável objetivo – cor recomendada.**

Expert SINTA		Intellec	
Branco	94,68	Branco	0,972

**Tabela 4.2 -- Resultado obtido para a segunda variável objetivo – doçura recomendada.**

Expert SINTA		Intellec	
Seco	43,2	Seco	0,432
		Suave	0,08

**Tabela 4.3 – Resultado obtido para a terceira variável objetivo – vinho.**

Expert SINTA		Intellec	
Chablis	38,857	Chablis	0,410
Sauvignon Blanc	32,721	Sauvignon Blanc	0,345
		Chardonay	0,072
		Soave	0,056

Analisando os resultados, as respostas se assemelham, embora o Intellec ofereça mais algumas opções como resposta. Isso se deve ao fato de que quando o Expert SINTA está verificando as premissas, se o valor fornecido pelo usuário para uma determinada variável é diferente do valor desta variável na premissa, a regra é descartada. No caso do Intellec, quando a resposta obtida é diferente da resposta existente, a premissa não é descartada, mas sim, o seu valor de verdade é assumido como 1 menos o valor de verdade fornecido pelo usuário.

Supondo-se que a seguinte premissa esteja sendo testada: *corpreferida=branco*. Ao ser perguntado sobre qual a cor de vinho preferida, o usuário responda tinto, mas com 0,7 de valor de verdade. O Intellec não rejeita a regra, mas passa a considerar, *corpreferida=branco* com 0,3 de valor de verdade, ou seja,  $1 - 0,7$ .

Esta consideração foi feita, pois se não há certeza absoluta de uma verdade, há alguma certeza de sua negação. Considerando-se que  $\text{certeza}(\text{tinto}) + \text{certeza}(\text{não tinto}) = 1$ , esta afirmação é válida para qualquer outra cor que não for tinto. O mesmo ocorre quando a variável deve ser inferida de outras regras.

Comparando os resultados obtidos através do Intellec e do Expert SINTA, notou-se uma diferença nos valores das variáveis. Isso se deve às diferentes fórmulas para a computação da verdade que cada máquina de inferência tem. No caso do Expert SINTA, a máquina de inferência calcula o grau de confiança da regra utilizando a Equação 4.1.

$$CFN(\text{regra}) = CFN(\text{conclusão}) * [CFN(\text{premissa1}) * CFN(\text{premissa2}) * \dots * CFN(\text{premissaN})]$$

Equação 4.1

onde CFN representa o grau de confiança.

- $CFN(\text{conclusão})$  é o grau de confiança da conclusão fornecido pelo especialista na montagem da base de conhecimento;
- $CFN(\text{premissa1})$ ,  $CFN(\text{premissa2})$ ,  $CFN(\text{premissaN})$  são os graus de confiança obtidos pela máquina de inferência para a premissa;
- $N$  é o número de premissas para a regra.

Já a máquina de inferência do Intellec, utiliza a Equação 4.2 para o cálculo do valor de verdade.

$$\mu(\text{regra}) = \mu(\text{conclusão}) \cdot \min[\mu(\text{premissa1}), \mu(\text{premissa2}), \dots, \mu(\text{premissaN})]$$

Equação 4.2

onde  $\mu$  representa o valor de verdade.

- $\mu(\text{conclusão})$  é o valor de verdade da conclusão fornecido pelo especialista na montagem da base de conhecimento;
- $\mu(\text{premissa1}), \mu(\text{premissa2}), \mu(\text{premissaN})$  são os valores de verdade obtidos pela máquina de inferência para a premissa;
- $N$  é o número de premissas para a regra.

Caso 2: Aqui, será implementado o mesmo caso do item anterior, mas as respostas dadas pelo usuário serão consideradas com valor de verdade igual a 1 para o Intellec e 100 para o Expert SINTA. Os resultados obtidos estão apresentados na Tabela 4.4, Tabela 4.5 e Tabela 4.6.

**Tabela 4.4 – Resultado obtido para a primeira variável objetivo – cor recomendada.**

Expert SINTA		Intellec	
Branco	97,2	Branco	0,972

**Tabela 4.5 – Resultado obtido para a segunda variável objetivo – doçura recomendada.**

Expert SINTA		Intellec	
Seco	48	Seco	0,48

**Tabela 4.6 – Resultado obtido para a terceira variável objetivo – vinho.**

Expert SINTA		Intellec	
Chablis	44,323	Chablis	0,456
Sauvignon Blanc	37,325	Sauvignon Blanc	0,384

Analisando os resultados qualitativamente, o Intellec oferece as mesmas respostas que o Expert SINTA. Isso ocorre porque neste exemplo, as regras que foram utilizadas para computação da verdade são as mesmas, visto que isso não ocorre no caso anterior,

pois a escolha foi feita com valor de verdade igual a 1. Isto é, quando se está testando a premissa *corpreferida=branco*, se o usuário responde tinto com 1 de valor de verdade, a premissa é rejeitada, da mesma forma que ocorre com o Expert SINTA.

Quanto à análise quantitativa dos resultados obtidos, o Intellec novamente apresentou uma pequena diferença nos valores das variáveis que pode ser atribuída ao uso das diferentes equações para a computação da verdade.

## **4.2 Sistema Especialista de Auxílio a Diagnóstico de Hepatites Virais**

O SE de auxílio a diagnóstico de hepatites virais está sendo desenvolvido pelo médico Manoel Tiago Vidal Ramos Jr., que é aluno de mestrado do Programa de Pós-Graduação de Ciências Médicas, da Universidade Federal de Santa Catarina.

As hepatites virais são doenças com alta prevalência na população brasileira. Dados do Ministério da Saúde consideram que 70% da população já teve contato com o vírus da hepatite A. No caso da hepatite B o contato é estimado em 15% com 1% da população sendo portadora deste vírus e 1,5% portadora do vírus C (TOLEDO, 2003). A Hepatite por vírus D acomete apenas pacientes já portadores do vírus B e, no Brasil, é endêmica na região amazônica (DANTAS *et al.*, 1995). A hepatite por vírus E pode causar doença grave em gestantes. A hepatite A, apesar da alta prevalência na população, não evolui para formas crônicas. Raramente um episódio de hepatite aguda evolui para hepatite fulminante, a maioria dos casos possuindo pouca manifestação clínica (isto é, não apresenta icterícia). As hepatites por vírus B e C podem tornar-se crônicas e, dentre esses casos, alguns evoluem para cirrose e carcinoma hepatocelular (DANTAS *et al.*, 1995, FRIDMANN e McQUAID, 2003).

O diagnóstico de hepatite aguda e o reconhecimento das formas crônicas podem auxiliar na prevenção da evolução para a cirrose. As hepatites virais são a segunda principal causa de cirrose no nosso meio, superadas apenas pelo alcoolismo e, em alguns países, são a principal causa de cirrose e indicação para o transplante hepático (DANTAS

*et al.*, 1995, FRIDMANN *et al.*, 2003).

A motivação para o desenvolvimento desse sistema especialista, especificamente, deve-se à complexidade da interpretação de exames em hepatites agudas e crônicas, à alta prevalência destas enfermidades na população e à dificuldade de acesso a especialistas em gastroenterologia, por parte da população brasileira.

A base de conhecimento utilizada pelo Intellec e, a utilizada pelo Expert SINTA encontram-se no relatório de pesquisa RP/M – IEB-UFSC – 01/2004.

Caso 1: O primeiro caso implementado foi um quadro típico clínico e laboratorial de um paciente adulto, sem fator de risco, com hepatite aguda por vírus B, que evoluiu para hepatite crônica B.

O quadro clínico é caracterizado principalmente pela presença de icterícia, náuseas, astenia, dor no abdômen superior e muitos episódios de vômitos. É observada também a ausência de febre e de emagrecimento. O paciente apresenta ingestão diária de etanol inferior a 20 gramas. Quanto ao exame físico, o paciente apresenta hepatomegalia dolorosa e consistência fibroelástica.

Os exames laboratoriais realizados neste paciente foram HBsAg (antígeno de superfície do vírus B), anti-HBcIgM (anticorpo para o vírus de centro do vírus de hepatite B), HBeAg (antígeno “e” da hepatite B) e HBV-DNA (pesquisa do ácido desoxirribonucleico do vírus da hepatite B) positivos, e o exame de ALT (alanina aminotransferase) apresenta seu valor acima de 10 vezes o limite superior.

Os resultados obtidos são mostrados na Tabela 4.7 e na Tabela 4.8.

**Tabela 4.7– Resultado obtido para a primeira variável objetivo – sugestivo de hepatite viral.**

Expert SINTA			Intellec		
Sugestivo Hepatite Aguda B	99,171		Sugestivo Hepatite Aguda B	0,991	
Sugestivo de Hepatite Viral	91,405		Sugestivo de Hepatite Viral	0,914	

**Tabela 4.8– Resultado obtido para a primeira variável objetivo – possibilidade diagnóstica final.**

Expert SINTA		Intellec	
Hepatite Crônica B	69,75	Hepatite Crônica B	0,697

Observando os resultados obtidos, verifica-se que os dois sistemas responderam de forma idêntica.

Caso 2: O segundo caso implementado foi um quadro típico clínico e laboratorial de um paciente adulto, com hepatite crônica C, com imunidade para o vírus B.

O quadro clínico é caracterizado principalmente pela presença de icterícia, náuseas, astenia, dor no abdômen superior e muitos episódios de vômitos. É observada também a ausência de febre e de emagrecimento. O paciente apresenta ingestão diária de etanol inferior a 20 gramas. Quanto ao exame físico, o paciente apresenta hepatomegalia dolorosa e consistência fibroelástica. O paciente realizou hemotransfusão antes de 1992.

Os exames laboratoriais realizados neste paciente foram anti-HCV (anticorpo para o vírus de hepatite C), HCV-RNA (pesquisa do ácido ribonucleico do vírus de hepatite C) e anti-HBs (antígeno para o HBsAg) positivos, e o exame de ALT apresenta seu valor de duas a quatro vezes o limite superior.

Os resultados obtidos são mostrados nas Tabela 4.9 e Tabela 4.10.



**Tabela 4.9 – Resultado obtido para a primeira variável objetivo – sugestivo de hepatite viral.**

Expert SINTA		Intellec	
Sugestivo de Hepatite Crônica C	92,916	Sugestivo de Hepatite Crônica C	0,929
Sugestivo de Hepatite Viral	76,125	Sugestivo de Hepatite Viral	0,761

**Tabela 4.10 – Resultado obtido para a primeira variável objetivo – possibilidade diagnóstica final.**

Expert SINTA		Intellec	
Imunidade vírus B	80	Imunidade Vírus B	0,8

Observando os resultados obtidos, verifica-se que os dois sistemas responderam de forma idêntica.

Analisando os resultados obtidos nestes dois últimos casos, verifica-se que os dois sistemas responderam de forma idêntica. Isso deve primeiramente ao fato deste exemplo apresentar primeiro, somente perguntas *Crisp*, ou seja, o valor verdade das variáveis são iguais a 1, e segundo, pelo fato das regras construídas pelo especialista possuírem, na maioria das vezes, regras com apenas uma premissa.

Para este exemplo foi consultada a opinião de um especialista do domínio, e os resultados apresentados pelo Intellec e pelo Expert SINTA se mostraram muito próximos ao da realidade.

Quanto ao DBIntellec, não foi realizado nenhum tipo de validação, visto que a implementação deste sistema não fazia parte do escopo do trabalho. Este sistema visa apenas auxiliar os profissionais da área da saúde a manter um cadastro de pacientes e um histórico de consultas. Estão sendo feitas ainda melhorias para deixá-lo mais aplicável.

## 5 Conclusões

O Intellec foi testado por 8 alunos do curso de graduação em Engenharia Elétrica da UFSC, na disciplina de Informática Médica e por 21 alunos da pós-graduação de Ciências Médicas da UFSC, na disciplina de Introdução à Informática Médica. Junto com a *shell* foi elaborado também um tutorial, para facilitar o seu entendimento e funcionamento.

As principais contribuições deste trabalho são: permitir ao especialista criar sistemas com várias metas, possibilitar uso de imagens como valores de variáveis e tutoriais como explicações para as perguntas. Sendo que as duas últimas características são as maiores vantagens do Intellec sobre o Expert SINTA.

O Expert SINTA apresenta uma característica que pode não ser interessante para alguns usuários: ele descarta uma regra se, ao verificar as premissas, o valor fornecido para a variável é diferente do valor apresentado por ela na premissa, como citado no primeiro exemplo do primeiro problema. Esta situação não ocorre no Intellec.

Outra diferença do Intellec sobre o Expert SINTA diz respeito ao uso de modelos *fuzzy* e não de fatores de certeza para o tratamento da incerteza, ou seja, é uma nova forma de montar uma base de conhecimento, sendo que, dependendo do problema em questão, o tratamento de incerteza por lógica *fuzzy* pode se mostrar mais adequado.

Pelo fato da máquina de inferência ter sido implementada como uma DLL, permitiu que fosse utilizada em conjunto com outra aplicação, escrita em uma linguagem de programação diferente.

O DBIntellec possibilita ao profissional da área da saúde criar um banco de dados de pacientes e histórico de consultas.

O Intellec já está sendo utilizado em trabalhos de pesquisa, em nível de mestrado, no Programa de Pós-Graduação de Ciências Médicas da UFSC (RAMOS *et al.*, 2004, OBERTO *et al.*, 2004).

Sugere-se como trabalhos futuros, melhorias no sentido de:

- Utilizar outras equações para obter o valor de verdade final para uma determinada conclusão.
- Estudar e implementar novas equações para o tratamento da incerteza.
- Permitir construir regras com conectivo “ou”. Isto diminuiria a quantidade de regras do sistema especialista.
- Possibilitar a construção de regras com várias conclusões, evitando assim, que tenha que se repetir um mesmo conjunto de premissas.
- Utilizar variáveis do tipo multivaloradas. Uma única variável pode receber vários valores em uma única consulta ao sistema.
- Apresentar um texto conclusivo da consulta realizada, além de apresentar quais as regras foram aceitas, para que o usuário possa ter um maior entendimento do porquê dos resultados obtidos.

## 6 Referências Bibliográficas

ACQUIRED INTELLIGENCE INC. *Acquire*. Disponível em: <<http://www.aiinc.ca>>. Acesso em: 22 de março de 2005.

ALGARVE, A. S., AZEVEDO, F. M., BARRETO, J. M.; 1995. *Prolog Implementation of a Shell for Developing Fuzzy Expert Systems*. In: ELECTRO'95 – XI CONGRESO CHILENO DE INGENIERIA ELECTRICA (13 a 17: Nov 1995: Punta Arenas, Chile). *Anais*. Chile, 1995. p. E012-E017.

ARARIBÓIA, G.; 1988. *Inteligência artificial: um curso prático*. Rio de Janeiro: Livros Técnicos e Científicos.

BARRETO, J. M.; 1997. *Inteligência Artificial – No Limiar do Século XXI*. Florianópolis: ppp Edições.

BITTENCOURT, G.; 2001. *Inteligência Artificial Ferramentas e Teorias*. Florianópolis: Editora da UFSC.

BRASIL, L. M.; 1994. *Aquisição de Conhecimento Aplicada ao Diagnóstico de Epilepsia*. Florianópolis. Dissertação (Mestrado em Engenharia Elétrica) - Centro Tecnológico, Universidade Federal de Santa Catarina.

CHORAFRAS, D. N.; 1988. *Sistemas Especialistas Aplicações Comerciais*. São Paulo: McGraw Hill.

CUNHA, F. S.; 1995. *Um Sistema Especialista para a Previdência Privada*. Florianópolis. Dissertação (Mestrado em Engenharia Elétrica) - Centro Tecnológico, Universidade Federal de Santa Catarina.

CUNHA H, RIBEIRO, S.; 1987. *Introdução aos Sistemas Especialistas*. Rio de Janeiro: Livros Técnicos e Científicos S.A.

DANTAS, W., MATTOS, A. A.; 1995. *Compêndio de Hepatologia*. São Paulo: Fundação Byk.

DE AZEVEDO, F. M., BRASIL, L. M. e DE OLIVEIRA, R. C.; 2000. *Redes Neurais com Aplicações em Controle e em Sistemas Especialistas*. Florianópolis: Visual Books.

DE AZEVEDO, F. M., FERRARI, G. L., ANGELONI, M. N. M., ARGOUD, F. I. M., ALGARVE, A. S.; 2004. *Shell para Desenvolvimento de Sistemas Especialistas Fuzzy usando Prolog*. In: III CONGRESSO LATINO AMERICANO DE ENGENHARIA BIOMÉDICA (Set. 2004: João Pessoa, Paraíba). *Anais*. Paraíba, 2004. p. 919-922

FERNANDES, A. M. R.; 2004 *Inteligência Artificial Aplicada à Saúde*. Florianópolis: Visual Books.

FISCHLER, M. A., FIRSCHEIN, O.; 1987. *Intelligence: The Eye, the Brain, and the Computer*. New York: Addison-Wesley.

FRIDMANN, S. L., McQUAID, K. R.; 2003. *Current Diagnosis & Treatment in Gastroenterology*. New York: Lange Medical Books.

GENARO, S.; 1987. *Sistemas Especialistas – O conhecimento Artificial*. Rio de Janeiro: Livros Técnicos e Científicos.

HAHNE, M. N. M.; 2001. *Implementação de um Shell para Desenvolvimento de Sistemas Especialistas Fuzzy usando Prolog*. Florianópolis. Dissertação (Mestrado em Engenharia Elétrica) - Centro Tecnológico, Universidade Federal de Santa Catarina.

HARMON, P., KING, D.; 1988. *Sistemas Especialistas*. Rio de Janeiro: Campus.

HEINZLE, R.; 1995. *Protótipo de um Ferramenta para Criação de Sistemas Especialistas Baseados em Regras de Produção*. Florianópolis. Dissertação (Mestrado em Engenharia Elétrica) - Centro Tecnológico, Universidade Federal de Santa Catarina.

IGNIZIO, J. P.; 1991. *Introduction to Expert Systems – The Development and Implementation of Rule-Based Expert Systems*. New York: McGraw Hill.

MONTELLO, M. V.; 1999. *Sistema Especialista para predição de Complicações Cardiovasculares integrado a um Sistema de Controle de Pacientes Portadores de Diabetes Mellitus*. Florianópolis. Dissertação (Mestrado em Engenharia Elétrica) - Centro Tecnológico, Universidade Federal de Santa Catarina

NOGUEIRA, J. H. M., ANDRADE E SILVA, R. B., ALCÂNTARA, J. F. L., HOLANDA, S. C., ANDRADE, R. C.; 1996. *Aplicações Baseadas no Expert Sinta, Uma ferramenta para a criação de Sistemas Especialistas*. Encontro de Iniciação Científica, UFC, 1996.

NOGUEIRA, J. H. M., ANDRADE E SILVA, R. B., ALCÂNTARA, J. F. L., HOLANDA, S. C., ANDRADE, R. C.; 1996. *Expert SINTA: Uma Ferramenta Visual Geradora de Sistemas Especialistas*. VI Semana de Informática, Universidade Federal da Bahia, 1996.

NORSYS SOFTWARE CORP. *NETICA*. Disponível em: <<http://www.norsys.com>>. Acesso em: 22 de março de 2005.

OBERTO, L. M., DE AZEVEDO, F. M.; 2004. *Sistema Inteligente de Auxílio ao Tratamento Fisioterápico aplicando o princípio da Neuroplasticidade em Portadores de Paralisia Cerebral*. In: IV WORKSHOP DE INFORMÁTICA APLICADA A SAÚDE - IV CONGRESSO BRASILEIRO DE COMPUTAÇÃO (Out. 2004 : Itajaí, Santa Catarina). *Anais*. Santa Catarina, 2004. p. 614-618.

RABUSKE, R. A.; 1998. *Inteligência Artificial*. Florianópolis: Editora da UFSC.

RAMOS, M. T. V., D'ACAMPORA, A. J., DE AZEVEDO, F. M.; 2004. *Sistema Inteligente de Auxílio ao Diagnóstico de Hepatites Virais*. In: IV WORKSHOP DE INFORMÁTICA APLICADA A SAÚDE - IV CONGRESSO BRASILEIRO DE

COMPUTAÇÃO (Out. 2004 : Itajaí, Santa Catarina). *Anais*. Santa Catarina, 2004. p. 595-598.

RICH, E.; 1988. *Inteligência Artificial*. São Paulo: Mc Graw Hill.

TOLEDO, A. C. J.; 2003. *Hepatites Virais – O Brasil está atento – Manuais Técnicos*. Brasília: Ministério da Saúde.

VIERA, A. F. G.; 1996. *Arandú: Um sistema Especialista Difuso para o Diagnóstico de Hepatopatias Crônicas*. Florianópolis. Dissertação (Mestrado em Engenharia Elétrica) - Centro Tecnológico, Universidade Federal de Santa Catarina.

WATERMAN, D.; 1986. *A Guide to Expert Systems*. USA: Addison-Wesley.