MARIA NAZARÉ MUNARI ANGELONI HAHNE

IMPLEMENTAÇÃO DE UM SHELL PARA DESENVOLVIMENTO DE SISTEMAS ESPECIALISTAS FUZZY USANDO PROLOG

FLORIANÓPOLIS
2001

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

IMPLEMENTAÇÃO DE UM SHELL PARA DESENVOLVIMENTO DE SISTEMAS ESPECIALISTAS FUZZY USANDO PROLOG

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

MARIA NAZARÉ MUNARI ANGELONI HAHNE

Florianópolis, Fevereiro de 2001

IMPLEMENTAÇÃO DE UM SHELL PARA DESENVOLVIMENTO DE SISTEMAS ESPECIALISTAS FUZZY USANDO PROLOG

Maria Nazaré Munari Angeloni Hahne

'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'

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

Prof. Almoraci da Silva Algarve, Dr.
Co-Orientador

Prof. Aguinaldo Silveira e Silva, Ph.D.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Prof. Fernando Mendes de Azevedo, D.Sc.
Presidente

Prof. Almoraci da Silva Algarve, Dr.

Prof. Renato Garcia Ojeda, Dr.

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

Prof. Raimes Moraes, Ph.D.

Aos meus pais, Alfredo e Helena.
Ao meu marido, Rafael.

AGRADECIMENTOS

Agradeço, primeiramente, ao meu orientador, Fernando Mendes de Azevedo, pela ajuda prestada no desenvolvimento deste trabalho, pelo incentivo e pela amizade.

Aos meus pais, Alfredo e Helena, que mesmo estando longe, me incentivaram, me ajudando a vencer mais esta etapa da minha carreira.

Ao meu marido, Rafael, que é também o meu melhor amigo, por ter estado ao meu lado em todos os momentos difíceis.

Aos meus irmãos, cunhados e sobrinhos, que me mostram sempre o verdadeiro sentido da palavra família.

Às amigas Ângela, Juliana, Lisiane, Moema, Tayze, Mônica, pelo convívio e bons momentos. Mais do que amigas, irmãs.

A Almoraci da Silva Algarve, que deu idéias muito importantes para o desenvolvimento deste sistema.

Ao colega de laboratório, Paulo Pizarro, por ter me mostrado caminhos no desenvolvimento do meu software.

A todos os colegas do Grupo de Pesquisas em Engenharia Biomédica.

A todas as pessoas, que, de alguma forma, contribuíram para o desenvolvimento deste trabalho.

À fundação CAPES, pelo apoio financeiro.

A Deus, por tudo que me dá todos os dias.

PUBLICAÇÕES

[1] Angeloni, M. N. M.; Azevedo, F.M. Fuzzy Expert System Shell Implemented Using Prolog. In: Chicago 2000 - World Congress on Medical Physics and Biomedical Engineering, Navy Pier, Chicago, IL, 23 a 28 julho de 2000. Abstract.

[2] Angeloni, M. N. M.; Azevedo, F.M, Algarve, A. S. Implementação de um Shell para Desenvolvimento de Sistemas Especialistas Fuzzy Usando Prolog. In: CBEB 2000 - XVII Congresso Brasileiro de Engenharia Biomédica, Florianópolis, 11 a 13 de setembro de 2000. p. 1124-1127.

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

IMPLEMENTAÇÃO DE UM SHELL PARA DESENVOLVIMENTO DE SISTEMAS ESPECIALISTAS FUZZY USANDO PROLOG

Maria Nazaré Munari Angeloni Hahne

Fevereiro / 2001

Orientador: Fernando Mendes de Azevedo, D.Sc.
Área de Concentração: Engenharia Biomédica
Palavras Chave: *Shell*, Sistemas Especialistas, *Prolog*
Número de Páginas: 132

Este trabalho apresenta a implementação de um *shell* para o desenvolvimento de Sistemas Especialistas *Fuzzy*. Este *shell*, implementado utilizando-se as linguagens *Delphi* e *Visual Prolog*, permite a implementação de um conjunto de regras *fuzzy*. A linguagem *Visual Prolog* é usada para implementar a máquina de inferência. A linguagem *Delphi* é usada para implementar a interface com o usuário. *Prolog* tem recebido uma grande atenção pela sua capacidade de resolução de problemas baseado na manipulação de proposições lógicas. *Prolog* é uma boa linguagem para programação lógica e seu estilo declarativo é uma de suas maiores características de interesse. A formulação de regras é construída de uma forma conversacional. A interface consiste de questões simples para implementar e escolher a variável *fuzzy* que represente a saída. O principal propósito deste *shell* é fornecer uma ferramenta simples para construir um sistema especialista usando lógica *fuzzy* para tratar a incerteza.

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

A FUZZY EXPERT SYSTEM SHELL IMPLEMENTED USING PROLOG

Maria Nazaré Munari Angeloni Hahne

February / 2001

Advisor: Fernando Mendes de Azevedo, D.Sc.
Area of Concentration: Biomedical Engineering
KeyWords: Shell, Expert Systems, Prolog
Number of pages: 132

This work presents a simple shell for the development of a Fuzzy Expert System. This shell allows the implementation of a set of fuzzy rules. It was implemented using Visual Prolog and Delphi languages. Visual Prolog language is used to implement the inference machine. *Delphi* language is used to implement the interface with the user. Prolog has received great attention by its capacity of problem solving based on the manipulation of logic propositions. Prolog is a good language for logic programming and its declarative style is a major feature of interest. The edition of rules occurs almost in a conversational form. The interface with the user consists of simple questions to implement and the choice of the fuzzy variable to show as the output. The main purpose of this shell is to provide a simple tool to build an Expert System using fuzzy logic to represent uncertainty.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

Os sistemas especialistas (SE) têm despertado grande interesse nos últimos anos. Isto se deve, principalmente, ao fato destes se constituírem em um dos poucos produtos concretos e viáveis dos estudos em Inteligência Artificial (IA). Eles têm sido utilizados em uma série de aplicações, indo desde sistemas de auxílio ao diagnóstico a sistemas de predição, por exemplo, no mercado de capitais.

Inúmeros fatores motivam a sua utilização. O principal deles, talvez, seja o custo e tempo necessários para formar especialistas humanos.

Inicialmente, os SE foram muito criticados devido a uma série de problemas. Entre eles, pode-se citar o problema da elicitação de conhecimento, da representação de conhecimento (por lógica, por regras de produção, por frames, entre outros), da construção de uma máquina de inferência e do tratamento da incerteza.

Por conseguinte, os SE não eram, nos seus primórdios, sistemas que apresentassem credibilidade. No entanto, os recentes avanços da tecnologia têm fornecido respostas às críticas mencionadas, criando ferramentas para solucioná-las. Hoje, existe um grande número de sistemas implementados fazendo uso de tais ferramentas em aplicações das mais diversas e passando por critérios rígidos de avaliação e validação. Devido a este fator, especialistas humanos estão confiando cada vez mais nestes sistemas, o que resulta em um número crescente de sistemas utilizáveis.

No entanto, o problema da dificuldade de implementação de um SE persistiu. Apenas especialistas em IA e SE eram e/ou são capazes de desenvolver estes sistemas.

Como já acontece em outros domínios da computação, fez-se necessário tornar a implementação de um SE cada vez mais facilitada para o usuário, dispensando estes da necessidade de conhecimentos profundos nas áreas mencionadas, de modo que cada vez mais pessoas sejam capazes de resolver estes problemas. Como exemplos de programas em outras áreas da computação que permitiram estas facilidades, pode-se citar programas para planilhas eletrônicas (*Excel*, *Lotus*), bancos de dados (*Access*, *Dbase*), programas de apresentação (*Powerpoint*). No caso de programadores e analistas, surgiram as linguagens visuais, disponibilizando ferramentas que implementam algumas das funções que são mais utilizadas no processo de desenvolvimento de programas. Como exemplos deste tipo de linguagem, pode-se citar o *Visual C* e *Delphi*.

No caso de SE, uma das linguagens que ficaram conhecidas na sua implementação foi o *Prolog*. Ao longo dos anos, foram surgindo novas versões desta linguagem, de forma a facilitar o trabalho do desenvolvedor. O *Turbo Prolog*, talvez, tenha sido o primeiro *Prolog* nesta direção. Hoje, conta-se com o *Visual Prolog*, que se encontra na versão 5.2. Contudo, mesmo com estes sistemas, desenvolvidos com o propósito de tornar o trabalho de implementação mais fácil, ainda existem muitas dificuldades. Cada vez mais, os sistemas exigem robustez e flexibilidade, o que implica, por sua vez, em maior dificuldade na tarefa de programar.

Um outro caminho encontrado para auxiliar no desenvolvimento de SE faz uso dos *shell*. Torna-se necessário dizer que um dos principais constituintes de um SE chama-se máquina de inferência, como será visto mais adiante. Por ocasião do desenvolvimento do MYCIN, foi observado que esta máquina de inferência poderia ser independente do problema tratado. Desta forma, podemos definir um *shell* como um programa que permite que sejam omitidos do usuário detalhes da implementação do sistema. No caso de *shell* para SE, a implementação da máquina de inferência fica transparente para o usuário.

Com estes programas, o usuário não precisa ser um profundo conhecedor de IA e de SE. Necessita-se de mais conhecimento das tarefas de elicitação e de representação do conhecimento (além de dispor da parceria de um especialista humano de domínio) com relação ao sistema que se deseja implementar. Os *shell* funcionam como ferramentas de IA orientadas para engenharia do conhecimento e construção de SE (RABUSKE, 1995).

A partir destas considerações, pode-se definir o objetivo do presente trabalho.

Objetivos: Implementar um *shell* para desenvolvimento de SE, utilizando a linguagem *Prolog* para implementar a máquina de inferência e lógica *fuzzy* para tratamento da incerteza.

Objetivos Específicos:

- Implementar um *shell* de fácil utilização permitindo, dessa forma, que o especialista de domínio possa desenvolver o sistema (sem a necessidade do engenheiro de conhecimento);
- Tratar a incerteza utilizando-se de conceitos da lógica *fuzzy*.

A necessidade de desenvolvimento deste *shell* surgiu das dificuldades encontradas em vários projetos desenvolvidos no Grupo de Pesquisas em Engenharia Biomédica (GPEB). Por um lado, muitos dos trabalhos de Dissertação de Mestrado ou Tese de Doutorado realizados no GPEB, na área de Informática Médica, utilizam-se de SE para auxílio ao diagnóstico em diferentes áreas da medicina. No entanto, tais trabalhos possuem outros objetivos que não apenas a construção do SE *per si*.

Por outro lado, em disciplinas ministradas no programa de Pós Graduação em Ciências Médicas, ao longo dos últimos oito anos, verificou-se o interesse cada vez maior dos médicos mestrandos por SE. No entanto, neste caso, a formação de base do público alvo (medicina) exigia que o ambiente de desenvolvimento do *shell* fosse o mais simples possível. Idealmente, sendo apenas necessária a implementação da base de conhecimento.

Em ambos os casos, uma solução encontrada pelos pesquisadores do GPEB foi a utilização do Expert SINTA, *shell* desenvolvido no Laboratório de Inteligência Artificial da Universidade Federal do Ceará (www.lia.ufc.br).

Paralelamente, o GPEB vem trabalhando para a implementação de tal *shell*. Primeiramente, ALGARVE (1995) desenvolveu um primeiro protótipo de tal sistema como exercício de uma disciplina de IA lecionada na Pós-Graduação em Engenharia Elétrica. Posteriormente, MONTELLO *et al.* (1997) implementaram uma versão simplificada deste primeiro protótipo, agora utilizando-se da linguagem *Visual Prolog*, também como trabalho da mesma disciplina.

No trabalho aqui apresentado, este *shell* foi desenvolvido. A linguagem *Prolog* foi escolhida para implementar a máquina de inferência. *Prolog* é uma linguagem que vem despontando como ferramenta de IA. Ela é uma linguagem declarativa, inicialmente concebida para processamento de linguagem natural, e da mesma forma que LISP, orientada para processamento simbólico, tendo bons recursos para processamento de listas. A atração que *Prolog* exerce, vem, em parte, do fato de ela ser diferente das linguagens tradicionais. O seu aspecto descritivo/declarativo faz com que "pensar a respeito do problema e aprender a programar em *Prolog* constituam um desafio intelectual excitante" (RABUSKE, 1995). *Prolog* tem, atrás de si, toda uma teoria matemática, a lógica dos predicados (RABUSKE, 1995).

A linguagem *Delphi* foi escolhida devido aos inúmeros recursos que ela

apresenta para a montagem de uma interface gráfica que torne o sistema bastante intuitivo para o usuário.

Os capítulos seguintes foram elaborados de modo a expor a teoria que fundamenta o trabalho e também mostrar o sistema que foi desenvolvido, com os resultado obtidos.

No capítulo dois, apresenta-se, primeiramente, o histórico, os conceitos, as aplicações referentes a SE, bem como alguns *shell* bastante conhecidos.

No capítulo três, discutem-se técnicas de tratamento de incerteza, tais como raciocínio probabilístico, fatores de certeza e lógica nebulosa ou *fuzzy*.

O capítulo quatro apresenta uma rápida introdução ao *Prolog*, mostrando, também, qual o mecanismo básico desta linguagem: o princípio da resolução. Neste mesmo capítulo, discute-se brevemente sobre raciocínio inferencial de forma a introduzir a fórmula de inferência chamada de *abduction*.

No capítulo cinco, introduz-se o sistema proposto, apresentando os seus módulos e a metodologia de desenvolvimento.

Finalmente, o capítulo seis apresenta dois protótipos de SE implementados neste *shell* e no Expert SINTA. Foram realizadas algumas consultas e as respostas das duas implementações comparadas qualitativamente, de modo a validar o sistema desenvolvido. Apresenta também sugestões de futuros melhoramentos.

2. SISTEMAS ESPECIALISTAS

Os SE constituem-se, talvez, no produto mais conhecido resultante dos estudos em IA, sendo uma das aplicações mais importantes, difundidas e utilizadas nesta área. O enfoque deste trabalho será para SE que se utilizam das técnicas da Inteligência Artificial Simbólica (IAS), visto que, atualmente, estão se desenvolvendo pesquisas, também, com SE híbridos, que utilizam tanto técnicas da IAS como da Inteligência Artificial Conexionista (IAC). Neste capítulo, será dada uma introdução a conceitos básicos de IA e seus dois paradigmas, para então detalhar o funcionamento dos SE.

2.1 Inteligência Artificial

Apesar de existirem inúmeras definições para IA, até hoje não se encontrou nenhuma que satisfaça todos os estudiosos da área. Todas elas são passíveis de críticas.

Segundo RICH *et al* (1994), "Inteligência Artificial é o estudo de como fazer os computadores realizarem coisas que, no momento, as pessoas fazem melhor".

Outra definição é de DE AZEVEDO (1993) que diz que ".. em uma primeira aproximação, pode-se considerar a IA como sendo o estudo de novos tipos de computadores e ferramentas, bem como métodos de solução de problemas para a implementação de novos sistemas que realizam tarefas que são consideradas inteligentes".

Poder-se-ia, também, dizer que o objetivo da IA é o de dotar o computador de um comportamento inteligente. Por comportamento inteligente, deve-se entender atividades que somente um ser humano seria capaz de efetuar. Dentro destas atividades podem ser citadas aquelas que envolvem tarefas de raciocínio (planejamento e estratégia) e percepção (reconhecimento de imagens, sons, etc.), entre outras (RABUSKE, 1995).

A história da IA é tão antiga quanto a própria civilização ocidental (ao que se saiba). Na realidade, ela começou com os estudos de Aristóteles a cerca dos processos de pensamento do ser humano, o que deu origem ao seu trabalho sobre Lógica,

aproximadamente 300 anos antes de Cristo. A Lógica de Aristóteles, cuja versão moderna é conhecida como Lógica dos Enunciados Categóricos, é a base de todas as lógicas modernas, algumas das quais fundamentais no desenvolvimento dos computadores (Lógica de Boole), bem como no desenvolvimento da IA (Lógica Proposicional e Cálculo dos Predicados). No entanto, apenas na década de 40 (século XX), devido principalmente aos esforços de guerra, surgiu o que se poderia, verdadeiramente, chamar de estudos de IA, em função das pesquisas em torno de seqüências de estratégia e análise do funcionamento do cérebro com objetivos de formalização de seu comportamento.

Inicialmente, os principais problemas tratados pela IA na sua busca pela reprodução de comportamento inteligente (na realidade o objetivo era o de resolução de problemas gerais) eram prova de teoremas e jogos. No entanto, a tarefa de resolução de problemas gerais se mostrou muito ambiciosa para o estado de desenvolvimento da época. Por conseguinte, as ambições e esperanças para a área foram diminuindo. Concluiu-se ser impossível para as máquinas da época (e de hoje também) a manipulação de um grande volume de conhecimento, como seria necessário para tratar de resolução de problemas gerais. Uma solução apontada foi tentar reduzir o máximo possível o campo de atuação do sistema em questão. Desta forma, surgiram os SE, onde a máquina deveria fazer inferências a partir de todo conhecimento que se pudesse adquirir acerca de um problema extremamente específico. Ou seja, os programas procuravam utilizar uma quantidade menor de conhecimento, realizando, dessa forma, tarefas bastante simples (RICH, 1994). Com o passar dos anos e com os avanços tecnológicos, a IA passou a manipular uma quantidade maior de conhecimento, apresentando assim grandes progressos nas áreas em que atuava (RICH, 1994).

BARRETO (1997) divide a história da IA nas seguintes fases:

- Época pré-histórica: até 1875, quando Camillo Golgi visualizou o neurônio.
- Época antiga: 1875 à 1943, com a publicação do trabalho de McCulloch e Pitts modelando o neurônio. Tinha como objetivo entender a inteligência humana.
- Época romântica: 1943 à 1956 (reunião no Darthmouth College). Nesta época, surgem os primeiros mecanismos imitando o funcionamento de redes de neurônios. Havia limitações devido à capacidade computacional.
- Época barroca: 1956 à 1969 (livro Perceptrons). Nesta época, pretendia-se propagar

o uso da IA, tanto usando a abordagem simbólica quanto a conexionista. Surgiram os primeiros SE usando a abordagem simbólica.

- Época das trevas: 1969 à 1981 (em outubro de 1981, os japoneses anunciaram seus planos para a Quinta Geração de Computadores). Os pesquisadores queriam encontrar aplicações práticas para a IA. A IA esbarrou, então, em interesses econômicos dos fabricantes, o que levou à perda de interesse nos estudos de novas aplicações.
- Renascimento: 1981 à 1987 (Primeira conferência internacional em redes neurais). Promoveu um renascimento da IA, tanto simbólica quanto conexionista. Surgimento do *Prolog*.
- Época contemporânea: 1987 até a atualidade. Neste período, há uma maior exploração das redes neurais artificiais (RNA's), devido a fatores como o melhoramento da capacidade de processamento da máquina. Inicia-se, também, a construção de sistemas chamados Sistemas Híbridos, que são a união das técnicas utilizadas nas duas linhas de pesquisa, permitindo a construção de sistemas que pretendem abranger uma forma mais completa de representação do comportamento humano.

Existem duas linhas ou paradigmas principais dentro do campo da IA: o paradigma simbólico ou IAS e o paradigma conexionista ou IAC:

- Inteligência Artificial Simbólica: Estudo da cognição, do raciocínio. Procura-se imitar o funcionamento do raciocínio humano na solução de um determinado problema, manipulando os conhecimentos básicos sem considerar o mecanismo responsável por isto (BARRETO, 1997). Este paradigma é interessante para resolver problemas bem definidos, onde tem-se o conhecimento sobre a maneira pela qual o problema é resolvido e que seja explícito o modo de achar a sua solução. Entre outros exemplos, pode-se citar jogos de xadrez, prova de teoremas, problemas de lógica, monitoração e diagnóstico. Para resolver problemas por meio deste paradigma, utiliza-se como ferramenta básica a Lógica, com suas regras de inferência inspiradas nos silogismos enunciados há mais de 2000 anos por Aristóteles (BARRETO, 1997).

- Inteligência Artificial Conexionista: Linha de pesquisa biológica, baseada em torno do funcionamento do cérebro e dos neurônios. Espera-se que, construindo um modelo que imite o funcionamento da estrutura do cérebro, este modelo apresente inteligência. Este paradigma, onde estão incluídas as RNA's, apresenta melhores resultados quando aplicado em problemas mal definidos, ou seja, quando não se tem o conhecimento explícito de como realizar determinada tarefa. A ferramenta básica para implementação destes sistemas é o complexo formado por circuitos se assemelhando à rede de neurônios cerebrais (BARRETO, 1997).

Ambos os paradigmas tentam resolver problemas relacionadas à busca, representação do conhecimento e aprendizagem. Estes dois paradigmas surgiram na década de 40 e estavam presentes em igualdade de condições no encontro ocorrido em 1956, no Darthmouth College. Este encontro foi o primeiro esforço conjunto para estudar IA (BARRETO, 1997). A corrente simbólica se acelerou, enquanto a corrente conexionista se manteve estacionada por muitos anos devido, primeiramente, à falta de recursos computacionais e, segundo, ao livro Perceptrons de MINSKY *et al* (1988). As pesquisas com redes neurais só foram retomadas anos mais tarde.

Os dois paradigmas possuem vantagens e desvantagens, e apresentam um melhor desempenho dependendo do tipo de problema ao qual são aplicados.

Entre as principais vantagens do paradigma simbólico, pode-se citar (BRASIL, 1999):

- Conhecimento *a priori* pode ser inserido diretamente;
- A seqüência de regras seguida pelo sistema para chegar a uma determinada conclusão pode ser mostrada ao usuário, servindo como uma explicação.

E como desvantagens, pode-se citar (BRASIL, 1999):

- Aprendizado não é uma de suas características intrínsecas;
- Dificuldade de construção (explicitação) da base de conhecimento;
- Fragilidade frente às informações aproximadas e incompletas;
- Processamento usualmente seqüencial e lento;
- Dificuldade no processo de Aquisição de Conhecimento (AC);

- Problemas no modo de representar o conhecimento de um dado domínio, por exemplo, por regras de produção, no sentido de determinar o número de regras necessárias para descrever o problema em questão, bem como o problema de conflito de regras e a atualização constante da base de conhecimento.

Como vantagens do paradigma conexionista, tem-se (BRASIL, 1999):

- A aquisição de conhecimento é fácil, permitida pelo aprendizado a partir de exemplos;
- Robustez das respostas devido à sua capacidade de generalização;
- Possibilidade de exploração do paralelismo. A recuperação das informações é extremamente rápida.

E as principais desvantagens do paradigma conexionista são (BRASIL, 1999):

- Dificuldade de determinação dos parâmetros de aprendizagem e da topologia da rede em função do problema;
- Conhecimento adquirido é codificado em pesos e ligações e de difícil interpretação (caixa preta), ou seja, não é uma tarefa fácil explicar o caminho seguido para chegar a uma conclusão. E isto implica na falta de interesse das indústrias pelas RNA's, já que os usuários querem saber o raciocínio por trás de uma conclusão obtida;
- Aprendizado dependente dos estados iniciais da rede.

Os SE Híbridos procuram aproveitar o que as duas abordagens oferecem de melhor, de modo a facilitar a implementação dos sistemas.

2.2 Sistemas Especialistas

OS SE[1] são sistemas computacionais projetados e desenvolvidos para solucionar um problema de forma semelhante a um especialista no seu domínio de conhecimento (BARRETO, 1997). Entende-se por especialista, aquela pessoa que, através de treinamento e experiência, alcançou um alto grau de conhecimento e competência em uma determinada área do conhecimento humano, sendo eficientes e exímios na sua área de atuação (MONTELLO, 1999).

Os primeiros trabalhos na área de SE surgiram por volta de 1960. No decorrer deste capítulo, serão citados alguns SE construídos e que se tornaram bastante conhecidos.

Para construir um SE eficiente, é necessário incorporar ao projeto não só um conjunto de informações, mas também a habilidade de manipulá-las de forma correta e eficiente. Outra característica interessante que um SE deve apresentar é a capacidade de aprender com a experiência e explicar o que está fazendo e porquê. Explicar o raciocínio é importante, pois permite ao usuário verificar a base de conhecimento, garantindo a sua confiança[2]. Dessa forma, o SE torna-se uma poderosa ferramenta de treinamento, instrução e educação. A explicação pode ser feita, por exemplo, através de um encadeamento para trás nas regras usadas no processo de inferência.

Hoje, existem inúmeras aplicações utilizando SE, nas mais diversas áreas de conhecimento humano. Como exemplos, citam-se (MONTELLO, 1999):

- Sistemas de Interpretação: deduz descrições a partir de observações, ou seja, faz a análise de fatos e procura relações entre estes e o seu significado. Por exemplo, os problemas de compreensão de fala, análise de imagens.
- Sistemas de Predição: deduz conseqüências a partir de situações. Permite uma previsão do futuro, baseado em dados passados e do presente. Por exemplo, os problemas de predição de tempo, clima e predição de tráfego.

---

[1] Inicialmente, a definição de Sistemas Especialistas confundia-se com a abordagem utilizada para implementá-los. Ou seja, eram confundidos com o paradigma simbólico. Hoje, distingue-se o conceito de SE da abordagem (DE AZEVEDO, 1993). Assim sendo, hoje, tem-se SE baseados em regras (os simbólicos), SE baseados em RNA's (os conexionistas) e SE híbridos.

[2] Na área médica, em particular, é assumido que um SE deva fornecer explicações do porquê de uma conclusão para ser aceito.

- Sistemas de Diagnóstico: deduz possíveis problemas a partir de observações ou sintomas. Por exemplo, os problemas de diagnóstico médico e mecânico.
- Sistemas de Projeto: desenvolve configurações de objetos que satisfazem determinados requisitos ou restrições. É um sistema capaz de justificar a alternativa tomada para o projeto final. Por exemplo, os problemas de projeto de circuitos digitais, projeto de edifícios.
- Sistemas de Planejamento: desenvolvem planos, cursos de ação para se atingir um determinado objetivo. Por exemplo, os problemas de movimento de robôs, estratégia militar ou comercial.
- Sistemas de Monitoração: devem verificar, de maneira contínua, um determinado comportamento de um sistema, intervindo, quando necessário, para alcançar objetivos com sucesso. Por exemplo, os problemas de monitoração de rede de distribuição elétrica, controle de tráfego aéreo.
- Sistemas de Depuração: prescreve correções para mau funcionamento provocado por distorções de dados. Por exemplo, problemas de depuração de programas.
- Sistemas de Reparo: desenvolvem e aplicam um plano para encontrar a solução para um determinado problema que foi diagnosticado. Por exemplo, problemas de manutenção de redes de comunicação, manutenção de sistemas de computação.
- Sistemas de Instrução: oferece mecanismos para verificação e ajuste do desempenho de estudante. Inclui toda a área de instrução assistida por computador (*computer-aided instruction*).
- Sistemas de Controle: controla o comportamento de um sistema (não apenas de computação) de forma adaptativa, ou seja, verificando os dados passados e fazendo uma predição do futuro. Por exemplo, problemas relacionados a robôs e gerência de produção.

Um SE apresenta como vantagem o fato de poder ser facilmente reproduzido, ou seja, várias cópias de um SE podem ser feitas e enviadas aos mais diversos lugares. Deste modo, o especialista artificial se torna bem mais disponível que o humano. Também possui a vantagem de fornecer sempre uma resposta rápida a uma consulta do usuário. Este tempo de resposta varia com a complexidade do conhecimento e com a

tecnologia utilizada para implementá-lo[3]. Este tipo de sistema sempre responde de acordo com as informações nele registradas, não estando sujeito a influências externas ou preferências.

Apresenta também desvantagens. O domínio de conhecimento coberto pelo SE é relativamente pequeno. Relações triviais são ignoradas pelo SE, a menos que elas tenham sido explicitadas na programação. A linguagem de comunicação com o usuário é limitada.

2.3 Estrutura de um sistema especialista

Os SE apresentam uma estrutura como a mostrada na Figura 2.1 (BARRETO, 1997). Estes são os blocos usuais de um SE implementado de modo simbólico.

A seguir, será dada uma breve descrição a respeito de cada bloco:

- Interface e explicação: É o módulo que contém a estrutura de diálogo que possibilita ao usuário acessar o sistema. Esta estrutura permite que ele responda às questões que estão sendo solicitadas pelo especialista; fornece uma resposta à consulta feita; mostra o encadeamento do raciocínio utilizado para a solução apresentada. Pode, também, fornecer ao usuário uma explicação de como o SE chegou à determinada conclusão.
- Base de conhecimento: É o conjunto de conhecimento a respeito do domínio do problema que será utilizado na tomada de decisões. As informações aqui presentes são obtidas do especialista de domínio. O conhecimento está representado de acordo com alguma técnica adequada ao sistema em questão e é fundamentalmente de dois tipos: fatos e regras que mostram como o especialista raciocina para chegar a uma conclusão.
- Motor de Inferência: Mecanismo de resolução de problemas. Opera a base de

[3] Por exemplo, um SE implementado usando-se uma RNA com topologia "feedforward" estática apresenta a resposta a qualquer questionamento no tempo necessário para a propagação da informação (qualquer que ela seja) ao longo dos neurônios das diversas camadas.

conhecimento, fornecendo ao usuário do sistema uma resposta de acordo com o quadro por ele apontado.

- Engenheiro de conhecimento: É o responsável pela criação da base de conhecimento. Ele obtém o conhecimento do especialista de domínio a partir de técnicas de elicitação de conhecimento. O conhecimento é transformado de forma conveniente e é adicionado à base de conhecimento. O especialista é a fonte de conhecimento do SE.
- Mecanismo de aprendizado: Quando este bloco é adicionado ao SE, ele oferece a capacidade de aprender. Com isto, a base de conhecimento que era estática, torna-se dinâmica, dando ao SE a capacidade de atualizar-se sem a intervenção do engenheiro de conhecimento.

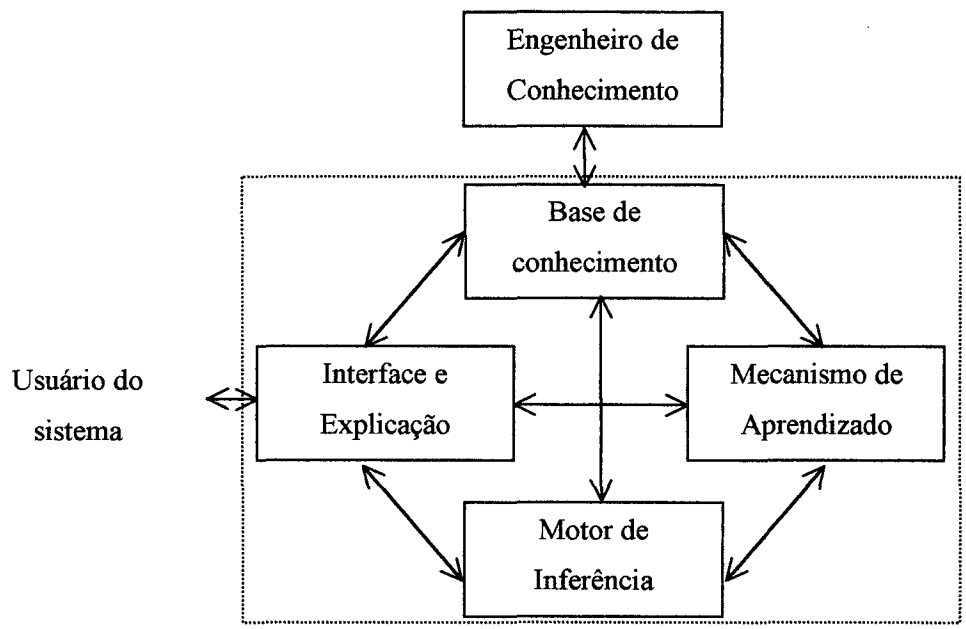

Figura 2.1 Blocos de um Sistema Especialista.

2.4 Diferenças entre Sistemas Especialistas e Programas Computacionais Convencionais

Os SE introduziram uma importante mudança no que diz respeito à filosofia de programação. Enquanto os programas computacionais convencionais lidam com algoritmos, ou seja, o programador define passo a passo o que o programa deve fazer, os SE lidam com conhecimento e processos de inferência, não sendo necessário informar os passos que ele deve executar para chegar a uma determinada resposta.

Desta forma, pode-se dizer que:

Dado + Algoritmo = Programa Computacional Convencional

Conhecimento + Inferência = Sistema Especialista

A Tabela 2.1 apresenta algumas das principais diferenças existente entre os SE e os programas computacionais convencionais (MONTELLO, 1999).

Tabela 2.1 Principais diferenças entre programas computacionais convencionais e SE.

Programas Computacionais Convencionais	**Sistemas Especialistas**
Representam e manipulam dados	Representam e manipulam conhecimento
Implementação de algoritmos	Implementação de heurística (ou regras)
Método de busca	Método de encadeamento
Modelagem do problema	Base de conhecimento
Possui analista (programador)	Possui engenheiro de conhecimento
Dificuldade de explanação (informação)	Facilidade de explanação (informação)
Relativa dificuldade de modificação	Facilidade de modificação

A partir disto, deve-se analisar se o problema em questão pode ser resolvido por um ou outro método, ou seja, considerar se o problema apresentado é um problema de IA ou um problema convencional. O problema de IA é aquele que é melhor resolvido por técnicas de IA, no caso, por SE. Por exemplo, um jogo de xadrez. É praticamente impossível, utilizando-se de programação convencional, resolver este problema. Da mesma forma, um programa que é facilmente resolvido por uma técnica convencional, pode se tornar trabalhoso de implementar se uma técnica de IA for utilizada.

2.5 Alguns sistemas especialistas

A década de 1980 foi marcada pelo grande crescimento de aplicações, inclusive com ampla disponibilização de produtos comerciais no mercado de *software*, devido principalmente aos avanços ocorridos com hardware neste mesmo período (MONTELLO, 1999).

Os SE destacam-se por terem sido os primeiros e mais avançados produtos comercialmente viáveis de IA (MONTELLO, 1999).

No decorrer destes anos, foram desenvolvidos vários SE. Alguns se tornaram marcos por apresentarem características que facilitaram a criação de novos sistemas. A seguir, citam-se alguns exemplos:

- DENDRAL: Construído em 1964 por Joshua Lederberg da Universidade de Stanford (EUA). É um programa do tipo algorítmico que, a partir de um determinado conjunto de dados como massa espectográfica e ressonância magnética, deduz a possível estrutura de um determinado composto químico (MONTELLO, 1999). Em 1965, Lederberg uniu-se a Edward Feigenbaum e Bruce Buchanan, com o intuito de construir um programa que usasse regras heurísticas para resolver os mesmos problemas do DENDRAL. Surge então, a idéia fundamental dos SE - a engenharia do conhecimento.

- MYCIN: Primeiro SE na área da medicina. Foi feito por Shortliffe no início dos anos 70. O programa recomenda a seleção de antibióticos em casos de bacteremia ou meningite, baseado em características do organismo infeccioso e em dados clínicos do paciente. Este sistema introduziu idéias que serviram como base para muitos dos SE em uso atualmente, entre elas (BARRETO, 1997):
  - Separação entre motor de inferência e base de conhecimento. Esta possibilidade foi comprovada com o programa EMYCIN, ou *Empty* MYCIN, que consiste basicamente no programa MYCIN sem sua base de conhecimento. Com isto, estava lançada a idéia de *shell* para desenvolvimento de um SE;
  - Um modo original de tratamento de incerteza: os fatores de certeza. MYCIN não somente sugeriu os fatores de certeza, como também mostrou que o raciocínio

médico não segue um padrão probabilístico.

Nesta mesma década, surgiram outros importantes e complexos sistemas especialistas como CADUCEUS, EXPERT, SOPHIE entre outros (MONTELLO, 1999).

- PROSPECTOR: SE criado no final dos anos 70 como apoio a geólogos na busca de depósitos com recursos geológicos. Foi o resultado do trabalho de Peter Hart e Richard Duda no Stanford Research Institute (BARRETO, 1997).

  Este programa seguiu muitas idéias introduzidas no MYCIN, tendo incorporado uma interface, chamada Lifer, permitindo a comunicação em linguagem técnica de geologia.

2.6 *Shell* para Sistemas Especialistas

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 os *shell* para SE.

Esta ferramenta facilita o trabalho de implementação de um SE, realizando muito do trabalho necessário para transpor um SE para um computador e permitindo seu uso por qualquer pessoa sem domínio profundo de IA, SE e informática. A idéia principal é separar a base de conhecimento (parte do sistema que trata especificamente do problema no domínio considerado) do motor de inferência (a parte que move o sistema, e que é comum). O usuário deve se preocupar apenas em obter o conhecimento do especialista humano. A máquina de inferência do sistema fica transparente ao usuário, ou seja, é inerente ao sistema.

Como vantagens em usá-los, pode-se citar:

- Prototipagem rápida de SE;
- Usam estruturas de dados e conhecimento pré-definidos (menos flexibilidade, mas mais rapidez e tranqüilidade para desenvolvedores);

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

Devido à facilidade que representam, existem vários *shell* disponíveis no mercado. Pode-se citar como exemplos:

- NETICA: Este software utiliza redes bayesianas para realizar vários tipos de inferência, usando algoritmos modernos e rápidos. Dado um novo caso, para o qual o usuário tem conhecimento limitado, NETICA encontrará os valores ou probabilidades apropriadas para todas as demais variáveis. O caso pode ser armazenado em um arquivo e depois incluído na rede (ou em uma rede diferente) para incrementar a consulta, ou para trazer uma nova informação sobre o caso. NETICA pode usar diagramas de influência para encontrar as decisões ótimas, que maximizam os valores esperados das variáveis especificadas (www.norsys.com).

- SPIRIT: É um *shell* para SE criado na Universidade de Hagen, na Alemanha. É uma ferramenta que possui interface gráfica de desenvolvimento. Permite a criação de diversos tipos de variáveis, tais como: booleana, ordinal e nominal. A parte qualitativa da rede de crença bayesiana é implementada a partir da inserção de regras de produção. Em seguida, valores de probabilidade são associados a estas regras e às variáveis, caracterizando a parte quantitativa da rede de crença bayesiana. Após a caracterização da rede bayesiana, é necessário inicializá-la, preparando-a para compilação, ou seja, a aprendizagem das regras. Feita a compilação, pode-se efetuar inferência sobre a base, inserindo evidências de um caso específico, chegando a um diagnóstico com um determinado valor de probabilidade. Este *shell* utiliza o conceito de entropia em seu algoritmo de cálculo da distribuição conjunta de probabilidade das variáveis em consideração (RODDER, 1996).

- Expert SINTA: Esta ferramenta foi desenvolvida pelo grupo SINTA (Sistemas INTeligentes Aplicados), integrante do Laboratório de Inteligência Artificial (LIA), do Departamento de Computação da Universidade Federal do Ceará (UFC) (www.lia.ufc.br). Utiliza um modelo de representação do conhecimento baseado em

regras de produção e fatores de certeza, simplificando o trabalho de implementação de SE através do uso de uma máquina de inferência compartilhada. Este *shell* será visto com mais detalhes na próxima seção.

2.7 Expert SINTA

A seguir, apresenta-se um resumo das características e funcionamento do Expert SINTA, baseado no documento *Manual do Usuário* (www.lia.ufc.br).

Além da máquina de inferência compartilhada, este *shell* permite a construção automática de telas e menus, o tratamento probabilístico das regras de produção e utilização de explicações sensíveis ao contexto da base de conhecimento modelada. O usuário responde a uma seqüência de menus, e o sistema encarrega se de fornecer respostas que se encaixam no quadro apontado pelo usuário.

Algumas características inerentes ao Expert SINTA são:

- utilização do encadeamento para trás;
- utilização de fatores de confiança;
- ferramentas de depuração;
- possibilidade de incluir ajudas *on-line* para cada base.

2.7.1 Arquitetura de um SE no Expert SINTA

Os SE gerados no Expert SINTA seguem a arquitetura mostrada na Figura 2.2, onde pode-se identificar:

- base de conhecimentos: representa o conjunto de informações (fatos e regras) que um especialista utiliza, representado computacionalmente;
- editor de bases: é o meio pelo qual o *shell* permite a implementação das bases desejadas;
- máquina de inferência: é a parte do SE responsável pelas deduções sobre a base de conhecimentos;

- banco de dados global: é o conjunto das evidências apontadas pelo usuário do SE durante uma consulta.

Uma base de conhecimento no Expert SINTA envolve os seguintes conjuntos de atributos que devem ser indicados pelo projetista da base: variáveis, regras, perguntas, objetivos, informações adicionais

Como padrão, o Expert SINTA grava as bases de conhecimento geradas em arquivos *.BCM.

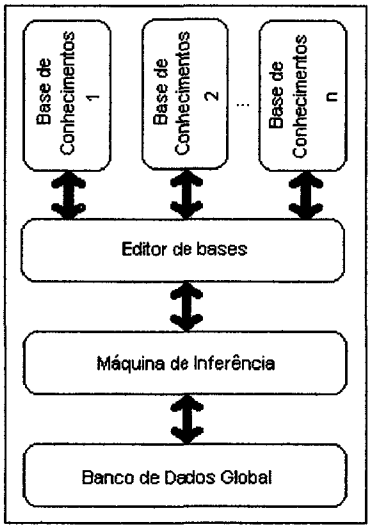

Figura 2.2 Arquitetura simplificada do Expert SINTA.

2.7.2 Criando Variáveis

Antes de se criar regras, é necessário que todas as variáveis utilizadas, bem como seus respectivos valores, sejam criados. Através desse mecanismo, a base fica organizada, fácil de manter e as regras podem ser criadas visualmente. Uma variável pode ser numérica, multivalorada ou univalorada

2.7.3 Definindo Objetivo(s)

O objetivo de uma consulta a um especialista é encontrar a resposta para um determinado problema. Em relação a um SE, funciona da mesma forma. A diferença é

que, aqui, os "problemas" são representados por variáveis. Antes de se poder executar o sistema pela primeira vez, é preciso que se definam quais são as variáveis (chamadas variáveis objetivo) que irão controlar o modo como a máquina de inferência se comporta.

2.7.4 Trabalhando com regras

O Expert SINTA utiliza regras de produção para modelar o conhecimento humano, o que o torna ideal para problemas de seleção, onde uma determinada solução deve ser atingida a partir de um conjunto de seleções.

Ao se definir as regras, a posição da regra em relação às demais deve ser indicada, pois a ordem influencia no descobrimento de soluções.

Para definir uma premissa, deve-se definir uma variável e seu valor.

Para inserir uma conclusão, procede-se de maneira análoga, escolhendo um item a partir do conectivo ENTÃO na sua regra. A última lista presente indica o grau de confiança daquela atribuição. Deixar a lista vazia indica uma confiança de 100% (cem por cento).

2.7.5 Definindo a Interface com o Usuário

Um SE implementado com o Expert SINTA comunica-se com o usuário final através de menus de múltipla escolha (ou escolha simples, se a variável em questão for univalorada). Estes menus são construídos automaticamente pelo *shell*, mas alguns detalhes devem ser fornecidos pelo criador da base. Por exemplo, a pergunta realizada pela máquina de inferência deve ser personalizada para que seja inteligível.

2.7.6 Consultando um especialista

Existem dois modos pelos quais se pode acompanhar uma consulta no Expert SINTA. Pode ser feita a execução normal ou então, realizar a depuração para verificar

passo a passo a execução da consulta, verificando valores de variáveis ou regras que estão sendo consultadas.

A consulta desenvolve se por meio de menus de múltipla (ou única) escolha. A escolha da(s) opção(ões) desejada(s) deve ser feita marcando as respectivas caixas de verificação, localizadas sempre à esquerda de cada alternativa. Existe a possibilidade de utilizar graus de confiança, que devem ser usados quando não se possui certeza absoluta sobre um fato.

3. TRATAMENTO DA INCERTEZA

Quando se considera problemas do mundo real, usualmente, trata-se com situações não booleanas. Ou seja, nos problemas reais, as situações (variáveis) não se apresentam simplesmente como "verdadeiras" ou "falsas". O mundo real é incerto. Incerteza, ou raciocínio aproximado ou inexato, segundo TURBAN (1992), "refere-se a uma larga faixa de situações onde a informação relevante é deficiente em um ou mais aspectos tais como ser parcial, não completamente confiável, imprecisa, conflitante, aproximada e sem uma relação causa/efeito absoluta".

Portanto, quando se constrói uma base de conhecimento e estabelecem-se os procedimentos de raciocínio, deve-se suplementar a representação do conhecimento e os métodos de inferência por modelos que manipulem (ou tratem) a incerteza. Alguns destes modelos são o objeto de estudo deste capítulo.

3.1 Representação de incerteza

Os três métodos básicos para representar a incerteza, ainda segundo TURBAN (1992) são o numérico, o gráfico e o simbólico, conforme detalhado a seguir.

- **Método Numérico**

Este é o método mais comum utilizado para representar a incerteza. Ele utiliza uma escala com dois números extremos. Um dos números representaria a incerteza completa, enquanto o outro extremo representaria a certeza completa.

Com este método, o especialista é solicitado a quantificar numericamente a sua confiança num determinado evento.

Embora pareça um método bastante simples, talvez por ser associado com probabilidades, apresenta alguns problemas para o especialista que deve quantificar estes valores. Como razão, pode-se citar o fato de a educação normalmente enfatizar um ambiente de certeza e a dificuldade que os especialistas têm em fornecer valores numéricos consistentes nas diferentes vezes em que lhe for solicitado (TURBAN,

1992). Por exemplo, muitas vezes um especialista da área médica encontra dificuldade em articular o conhecimento usado para obter um diagnóstico, como também em explicar o porquê do diagnóstico.

➢ **Método Gráfico**

Este método pode ser utilizado para auxiliar especialistas que tenham dificuldades em expressar a confiança em um determinado evento em termos numéricos.

O método gráfico pode ser utilizado com ou sem números. A Figura 3.1(a) mostra um gráfico de barras sem números, onde o especialista deve ser solicitado a marcar algum lugar da escala, mostrando sua maior ou menor confiança em um evento. As Figuras 3.1(b) e 3.1(c) mostram gráficos com números. A Figura 3.1(b) mostra uma apresentação de confiança tipo probabilidade. A Figura 3.1(c) mostra uma escala centrada em zero.

Este método não é tão utilizado quanto o método numérico.

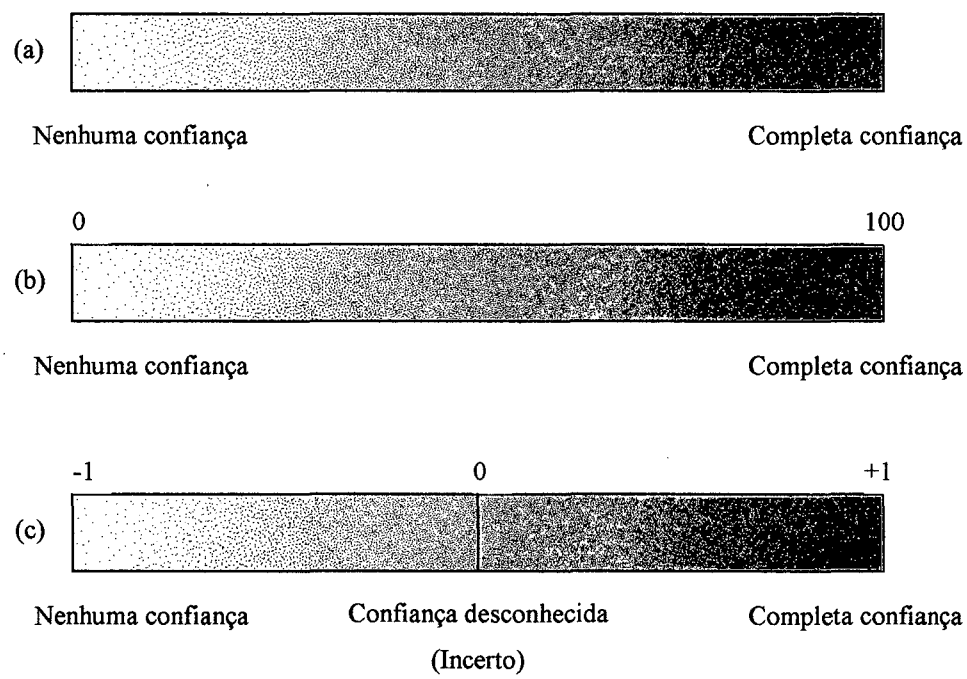

Figura 3.1 Escalas de confiança (a) Escala sem números, (b) Escala tipo probabilidade e (c) Escala centrada em zero.

➢ **Método Simbólico**

Este não se constitui em um, mas em vários métodos diferentes que se utilizam de símbolos para que o especialista possa representar sua confiança em um evento. Entre eles pode-se citar:

- *Likert scale*: O especialista define uma escala de termos lingüísticos para representar a incerteza. Por exemplo, a temperatura da água pode ser fervendo, quente, morna, fria, gelada.
- *Ranking*: Pode ser utilizado na forma ordinal, isto é, listando itens pela sua importância, ou então cardinal, isto é, o *ranking* é complementado por valores numéricos.

Este tipo de representação é freqüentemente associada a números ou, então, convertido para valores numéricos.

No caso dos SE, vários modelos de tratamento da incerteza podem ser utilizados. A seguir, os seguintes modelos serão descritos:

- Raciocínio Probabilístico;
- Fatores de Certeza;
- Lógica Nebulosa ou *Fuzzy*. Os termos f*uzzy* e nebuloso serão utilizados indistintamente.

3.2 Raciocínio Probabilístico

Este é, talvez, um dos métodos mais antigos para tratamento de incerteza (BARRETO, 1997). Neste tipo de raciocínio, expressa-se o grau de confiança em fatos de um domínio como uma probabilidade (TURBAN, 1992). A conclusão obtida a partir destes fatos também está associada a uma probabilidade. Como probabilidade, pode-se definir a chance que um determinado evento tem de ocorrer. Esta chance, agora probabilidade, pode assumir qualquer valor real no intervalo [0,1].

Formalmente, a probabilidade de que um evento X ocorra é dada pela fórmula:

$$P(X) = \frac{\text{Número de resultados possíveis em que X occore}}{\text{Número total de resultados possíveis}} \tag{3.1}$$

Pode-se, também, associar as probabilidades de ocorrência de diversos eventos através dos conectivos lógicos *e* ($\wedge$) e *ou* ($\vee$).

No caso da probabilidade condicional, expressa por $P(A/B)$, a probabilidade de ocorrência de um evento A ("conseqüência") dada a ocorrência de um outro evento B ("causa"), pode ser interpretada como o grau de crença de que A é verdadeiro dado B. Assim tem-se:

- Se $P(A/B) = 1$, logo, a crença é de que A é verdadeiro, quando B é verdadeiro;
- Se $P(A/B) = 0$, a crença é de que A é falso, quando B é verdadeiro;
- Se $0 < P(A/B) < 1$, para outros valores, isto significa que não se está inteiramente seguro que A é verdadeiro ou falso, quando B é verdadeiro.

Quando há probabilidade condicional, usa-se o teorema de Bayes para o seu cálculo. Considerando-se que H é uma hipótese e E é uma evidência, a probabilidade $P(H/E)$, segundo o teorema de Bayes, significa a probabilidade condicional de que a hipótese H ocorra, dado que a evidência E ocorreu. Formalmente, é dado por (NASSAR, 1998):

$$P(H/E) = \frac{P(E/H) * P(H)}{P(E)} \tag{3.2}$$

onde,

- $P(E/H)$ é a probabilidade de E ocorrer dado que H ocorreu;
- $P(H)$ é a probabilidade de H ocorrer isoladamente;
- $P(E)$ é a probabilidade de E ocorrer isoladamente.

No entanto, a fórmula 3.2 contempla apenas o caso mais simples, onde tem-se apenas uma hipótese e uma evidência.

Então, vamos supor que se tenha um conjunto H de hipóteses. Pode-se provar que, para uma seqüência H_i exaustiva e mutuamente exclusiva dois a dois, a probabilidade de ocorrência de H_i dada a evidência E é dada por (NOLT, 1991):

$$P(H_i / E) = \frac{P(E / H_i) * P(H_i)}{\sum_{n=1}^{k} P(E / H_n) * P(H_n)} \tag{3.3}$$

onde,

- $P(H_i / E)$ é a probabilidade de que a hipótese H_i seja verdadeira, dada a evidência E;
- $P(E / H_i)$ é a probabilidade de que se observe a evidência E, dado que a hipótese H_i é verdadeira;
- $P(H_i)$ é a probabilidade, *a priori*, de que a hipótese H_i seja verdadeira na ausência de qualquer evidência específica;
- k é o número de hipóteses possíveis.

Este teorema pode ser atualizado quando ocorre uma nova evidência. Seja H uma hipótese, $E = e_1, e_2, ..., e_n$ uma seqüência de dados independentes observados no passado e seja e um novo fato. Utiliza-se a seguinte fórmula (NASSAR, 1998):

$$P(H / E, e) = P(H / E) * \frac{P(e / E, H)}{P(e / E)} \tag{3.4}$$

Porém, é comum termos várias hipóteses concorrentes, ou então, multivaloradas. Para o caso de hipóteses multivaloradas, a crença na i-ésima hipótese H_i, supondo-se um conjunto de n evidências e m hipóteses, é dado por (NASSAR, 1998):

$$P(H_i / e_1, e_2, e_p, ..., e_n) = \frac{P(e_1, e_2, ..., e_p, ..., e_n / H_i) * P(H_i)}{P(e_1, e_2, ..., e_p, ..., e_n)} \tag{3.5}$$

Denominando $[P(e_1, e_2, ..., e_p, ..., e_n)]^{-1}$ como constante de normalização α, e considerando a independência das evidências com relação a cada hipótese H_i, então, a fórmula 3.5 torna-se (NASSAR, 1998):

$$P(H_i / e_1, e_2, e_p, ..., e_n) = \alpha * P(H_i) * \prod_{p=1}^{n} P(e_p / H_i) \quad (3.6)$$

Note que:

- $P(H_i)$ é o vetor de probabilidades *a priori* das diferentes hipóteses em consideração;
- A constante de normalização α deve ser calculada pela restrição da soma unitária das probabilidades $P(e_1, e_2, ..., e_p, ..., e_n)$.

Assumindo ainda que para a evidência e_p, define-se o vetor λ_p de probabilidades condicionais das m hipóteses, isto é, $\lambda_p = (\lambda_1, \lambda_2, ..., \lambda_m)$, onde, $\lambda_i = P(e_p / Hi)$

Finalmente, pode-se escrever a equação 3.5 na seguinte forma (NASSAR, 1998):

$$P(H_i / e_1, e_2, e_p, ..., e_n) = \alpha * P(H_i) * \Delta \quad (3.7)$$

onde,

- $\Delta = (\lambda_1 * \lambda_2 * ... * \lambda_n)$.

Assim, fica computacionalmente mais fácil obter o vetor de probabilidades condicionais das hipóteses H_i dado um conjunto de evidências.

O uso do raciocínio probabilístico envolve a obtenção da distribuição conjunta de probabilidade de variáveis aleatórias, que leva à necessidade de informações sobre as relações entre estas variáveis. No pior caso, necessita-se das probabilidades de todas as combinações possíveis de variáveis aleatórias, fato este que levaria a uma explosão combinatória.

E este fato torna impraticável o uso do Teorema de Bayes por ser necessário um grande espaço para armazenar probabilidades, grande gasto de tempo para computar todas as probabilidades e necessidade de informação de um grande número de probabilidades.

Mesmo com estes problemas, as estatísticas bayesianas são uma base atraente para um raciocínio sob incerteza. Esta teoria é adequada para situações bem estruturadas em que todos os dados são disponíveis. Infelizmente, esta situação não ocorre muito

freqüentemente. O procedimento não pode tratar com representação de conhecimento qualitativo ou com ignorância. Existem também dificuldades com a obtenção de probabilidades *a priori* bem como de novas evidências (TURBAN, 1992).

E para solucionar os problemas mencionados, foram criadas outras técnicas para seu tratamento, entre elas, as redes bayesianas.

Em uma rede bayesiana, ao invés de se explicitar a distribuição conjunta de probabilidades, são explicitadas tabelas de probabilidades condicionais que por sua vez podem ser utilizadas para obter as distribuições conjuntas.

Este tipo de rede é utilizada para representar o conhecimento em um SE probabilístico.

As redes bayesianas são compostas de duas partes: uma qualitativa e outra quantitativa. A parte qualitativa é um modelo gráfico (grafo acíclico direcionado) onde as variáveis são os nós e as regras, que são as relações de dependência entre essas variáveis, são os arcos direcionados. Assim, um arco ligando A e B na seguinte forma $A \rightarrow B$, indica que a variável B é a conseqüência e a variável A é a causa; estas representam uma relação de dependência. Por outro lado, se não houver um arco ligando duas variáveis, assume-se que estas são independentes.

A parte quantitativa de uma rede bayesiana é o conjunto de probabilidades condicionais associadas aos arcos existentes no modelo gráfico acima descrito e as probabilidades estimadas *a priori* das hipóteses diagnósticas.

Atualmente existem *shell* para o desenvolvimento de SE com raciocínio probabilístico (redes Bayesianas). Pode-se citar o SPIRIT (RODDER, 1996) e o NETICA (www.norsys.com).

Na prática, no entanto, torna-se difícil prover os SE com probabilidades verdadeiras, em vista da dificuldade de determiná-las, conforme observado nos parágrafos acima. Por conseguinte, hoje em dia discute-se a probabilidade subjetiva, onde se trata eventos que não têm uma base histórica (base de dados) sobre a qual se possa extrapolar. A probabilidade subjetiva é realmente uma crença ou opinião expressa como uma probabilidade mais do que uma probabilidade baseada em medidas empíricas. Ou seja, nos SE probabilísticos, os valores de probabilidade refletem a crença do especialista sobre o que ele espera que ocorra em situações similares àquelas que ele tem experimentado (NASSAR, 1998).

3.3 Fatores de Certeza

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

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 mesma evidência E, sendo representado por:

$$FC(H,E) = MC(H,E) - MD(H,E) \tag{3.8}$$

onde,

- $FC(H,E)$ é o fator de certeza na hipótese H dada a 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 MC e MD são, por sua vez, dadas por (RICH, 1988):

$$MC(H,E) = \begin{cases} 1 \text{ se } P(H) = 1 \\ \dfrac{\max[P(H/E), P(H)] - P(H)}{\max[1,0] - P(H)}, \text{ em outros casos} \end{cases}$$

$$MD(H,E) = \begin{cases} 1 \text{ se } P(H) = 0 \\ \dfrac{\min[P(H/E), P(H)] - P(H)}{\min[1,0] - P(H)}, \text{ em outros casos} \end{cases} \tag{3.9}$$

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.

Uma determinada evidência aumenta a probabilidade de H e, neste caso,

$MC(H,E)>0$ e $MD(H,E)=0$, ou diminui a probabilidade de H sendo que, neste caso, $MC(H,E)=0$ e $MD(H,E)>0$, conforme pode ser deduzido das fórmulas mostradas em 3.9. Isto implica que o FC pode ser positivo ou negativo. Se o FC for positivo, o sistema acredita que a hipótese é verdadeira, e se FC for negativo, o sistema acredita que a hipótese é falsa.

O FC difere da teoria da probabilidade. Uma das diferenças, talvez a principal, é o fato de que, na teoria da probabilidade, a soma das probabilidades é sempre igual a 1. Isto não ocorre com os fatores de certeza. Quando se afirma que a possibilidade de dar sol amanhã é igual a 0,75, não implica necessariamente em se afirmar que a possibilidade de não dar sol seja igual a 0,25. No caso da teoria da probabilidade, esta segunda probabilidade seria, obrigatoriamente, igual a 0,25 (RICH, 1994).

A Tabela 3.1 (BARRETO, 1997) mostra algumas características referentes aos FC, MC e MD.

Tabela 3.1 Algumas características de medidas de crença, descrença e fatores de certeza.

Características	Valores
Variações	$0 \le MC \le 1$ $0 \le MD \le 1$ $-1 \le FC \le 1$
Certeza das hipóteses verdadeiras $P(H/E) = 1$	$MC = 1$ $MD = 0$ $FC = 1$
Certeza das hipóteses falsas $P(\neg H/E) = 1$	$MC = 0$ $MD = 1$ $FC = -1$
Perda da Evidência $P(H/E) = P(H)$	$MC = 0$ $MD = 0$ $FC = 0$

3.3.1 Combinando fatores de certeza

Os fatores de certeza, da mesma forma que as probabilidades, precisam ser

combinados quando duas ou mais evidências existem. Neste caso, considerando-se uma hipótese H e duas evidências e_1 e e_2, MC e MD passam a ser dadas por (RICH, 1988):

$$\begin{aligned} MC(H, e_1 \wedge e_2) &= \begin{cases} 0 \text{ se } MD(H, e_1 \wedge e_2) = 1 \\ MC(H, e_1) + MC(H, e_2) * (1 - MC(H, e_1)), \text{ em outros casos} \end{cases} \\ MD(H, e_1 \wedge e_2) &= \begin{cases} 0 \text{ se } MC(H, e_1 \wedge e_2) = 1 \\ MD(H, e_1) + MD(H, e_2) * (1 - MD(H, e_1)), \text{em outros casos} \end{cases} \end{aligned} \tag{3.10}$$

Para exemplificar como este mecanismo funciona, seja uma observação inicial que confirme a crença em H com MC = 0,3. Então, concluí-se que $MD(H, e_1) = 0$ e $FC(H, e_1) = 0{,}3$. Agora, faz-se uma segunda observação, que também confirma H, com $MC(H, e_2) = 0{,}2$. Então,

$$MC(H, e_1 \wedge e_2) = 0{,}3 + 0{,}2 * 0{,}7 = 0{,}44$$

$$MD(H, e_1 \wedge e_2) = 0$$

$$FC(H, e_1 \wedge e_2) = 0{,}44$$

Faz-se, também, necessária a combinação de crença quando existem várias hipóteses. Da mesma maneira, para o caso de combinação de duas hipóteses h_1 e h_2, dada uma evidência E, as medidas de crença são dadas por (RICH, 1988):

$$\begin{aligned} MC(h_1 \wedge h_2, E) &= \min(MC(h_1, E), MC(h_2, E)) \\ MC(h_1 \vee h_2, E) &= \max(MC(h_1, E), MC(h_2, E)) \end{aligned} \tag{3.11}$$

sendo que MD pode ser calculada de maneira análoga.

Uma característica importante desta abordagem é que ela parece imitar muito bem a maneira como as pessoas manipulam a incerteza (RICH, 1994).

A abordagem faz suposições de independência que tornam seu uso relativamente fácil. Ao mesmo tempo, deve-se ter cuidado na elaboração das regras para que elas explicitem dependências importantes.

A maior vantagem dos FC é a fácil computação pela qual a incerteza é propagada no sistema. Os FC são fáceis de compreender e separam claramente crença de descrença (BARRETO, 1997).

A grande dificuldade consiste, da mesma forma que no modelo probabilístico, na necessidade de determinar as probabilidades de forma a se obter os FC. Ou seja, para muitas hipóteses e evidências, o grande número de probabilidades a ser determinado inviabiliza a adoção do procedimento.

A solução encontrada é trabalhar-se com valores de FC atribuídos pelo especialista de domínio com base em sua experiência. O fato da soma dos FC não ter necessariamente que ser igual a 1, como na teoria da probabilidade, facilita esta assunção por parte do engenheiro de conhecimento e do especialista de domínio, quando da construção de um SE.

3.4 Raciocínio Nebuloso

A Lógica *Fuzzy* (ou lógica nebulosa) foi proposta por Lotfi Zadeh em 1965, como uma extensão dos conjuntos clássicos (SANDRI, 1999).

Enquanto a teoria tradicional define que um dado valor pertence ou não a um determinado conjunto com um valor booleano (pertence ou não pertence), na teoria *fuzzy*, o elemento pode pertencer, não pertencer ou estar parcialmente presente em um determinado conjunto, segundo um valor ou grau de pertinência (RICH, 1988). Este valor de pertinência pertence a uma faixa de 0 (elemento não pertencente ao conjunto) até 1 (elemento totalmente pertencente ao conjunto). A relação entre os valores de um elemento e seu grau de pertinência em um conjunto é feita por uma função de pertinência, ou seja, a função de pertinência mapeia entradas numéricas a graus de pertinência.

Por exemplo, ao considerar a variável *temperatura*. Utilizando-se deste método, pode-se dividí-la dentro de uma faixa de estados: *frio* (*T0* a *T2*), *fresco* (*T1* a *T3*), *moderado* (*T2* a *T4*), *morno* (*T3* a *T5*), *quente* (*T4* a *T6*), conforme a Figura 3.2. Seta-se um valor arbitrário para dividir os estados.

O estado da variável de entrada não salta abruptamente de um estado para o próximo; ao invés disso ele perde gradualmente valor em um estado enquanto vai ganhando valor no próximo estado. O valor de temperatura é quase sempre algum ponto entre duas funções consecutivas: 0,6 moderado e 0,4 morno, ou 0,7 moderado e 0,3 fresco, e assim por diante.

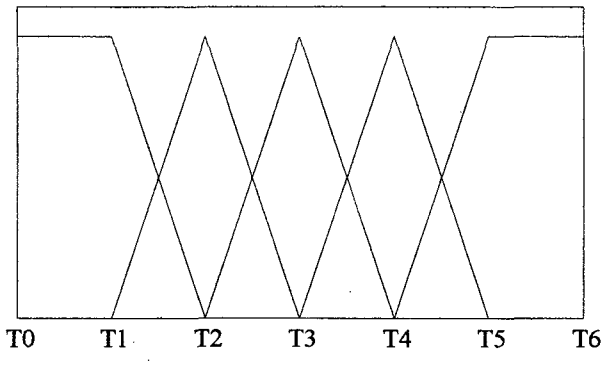

Figura 3.2 Função de pertinência para temperatura.

Como mencionado anteriormente, utilizam-se funções de pertinência para obter o grau de pertinência de um valor em um determinado conjunto. Pode-se citar, entre as funções mais utilizadas (SANDRI, 1999):

- Triangular: é especificada por três parâmetros $\{a,b,c\}$, que determinam a coordenada x dos três cantos do triângulo, como mostrado na Figura 3.3(a). E é dada pela fórmula:

$$Triângulo(x,a,b,c) = \max(0, \min[(x-a)/(b-a), (c-x)/(c-b)]) \quad (3.12)$$

- Trapezoidal: é especificada por quatro parâmetros $\{a,b,c,d\}$, como mostrado na Figura 3.3(b). E é dada pela fórmula:

$$Trap(x,a,b,c,d) = \max(0, \min[(x-a)/b-a), 1, (d-x)/(d-c)]) \quad (3.13)$$

- Gaussiana: é especificada por dois parâmetros $\{s,c\}$, como mostrado na Figura 3.3(c). E é dada pela fórmula:

$$gaussiana(x,s,c) = \exp\{-(x-c)/s\}2 \quad (3.14)$$

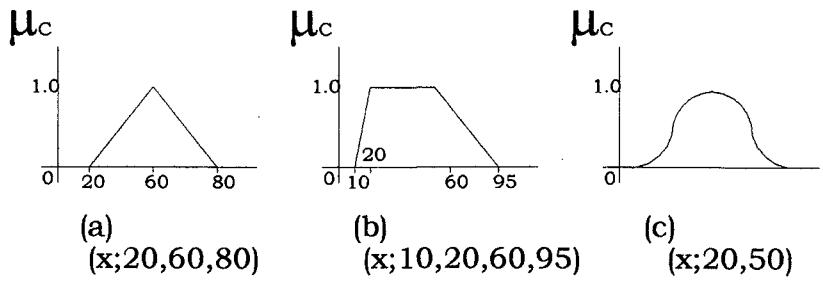

Figura 3.3 Tipos de funções de pertinência: (a) Função triangular, (b) Função trapesoidal, (c) Função gaussiana.

Matematicamente, sendo x um elemento pertencente ao universo de discurso X, um conjunto *fuzzy* A em X é definido como um conjunto de pares ordenados:

$$A = \{(x, \mu_A(x)) / x \in X\}$$

onde $\mu_A(x)$ é chamada de função de pertinência de x em A. A função de pertinência mapeia cada elemento de X com um valor de pertinência contínuo entre 0 e 1.

Normalmente, os valores de pertinência são definidos para variáveis lingüísticas (ex.: temperatura alta) através da quantificação da percepção humana. A variável lingüística é caracterizada por uma quíntupla (x, $T(x)$, U, G, M), onde:

- x é o nome da variável;
- $T(x)$ é o conjunto de termos de x, que é o conjunto de nomes dos valores lingüísticos de x no qual cada valor é um conjunto *fuzzy* definido em U;
- U é o universo de discurso, que representa todos os valores possíveis de uma variável do sistema;
- G é a regra sintática para gerar o nome dos valores de x;
- M é a regra semântica para associar cada valor com seu significado.

Por exemplo, se a velocidade de um automóvel for interpretada como uma variável lingüística, então seus conjuntos de termos T(velocidade) poderia ser:

$$T(\text{velocidade}) = \{\text{lenta, média, rápida}\}$$

Cada termo em T(velocidade) é caracterizado por um conjunto *fuzzy* no universo de discurso $U = [0,150]$. O termo lento pode ser interpretado como a velocidade abaixo de 60 Km/h, o termo médio como a velocidade em torno de 80 Km/h e rápido como a velocidade acima de 120 Km/h. Esses termos podem ser caracterizados como um conjunto *fuzzy*, no qual as funções de pertinência são mostradas na Figura 3.4.

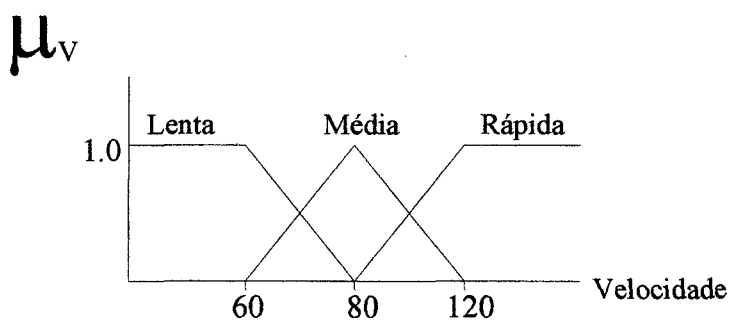

Figura 3.4 Função de pertinência para velocidade de um automóvel.

É possível realizar diversas operações utilizando conjuntos *fuzzy*. Considere-se A e B dois conjuntos *fuzzy* em U, com suas funções de pertinência μ_A e μ_B, respectivamente. As operações de conjuntos *fuzzy*, como união, intersecção e complemento, são definidas através de suas funções de pertinência (SANDRI,1999).

- União: A função de pertinência $\mu_{A\cup B}(x)$ (Figura 3.5) é definida ponto a ponto para todo $x \in U$ por:

$$\mu_{A\cup B}(x) = \mu_A(x) \vee \mu_B(x) \quad \text{ou} \quad \mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

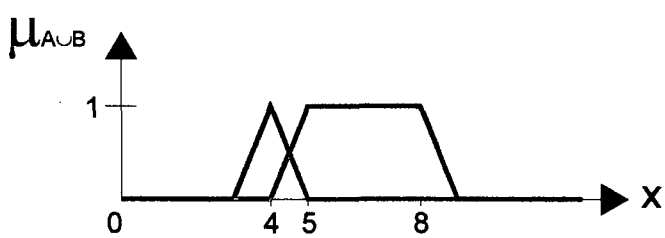

Figura 3.5 Operação do conjunto união.

- Intersecção: A função de pertinência $\mu_{A \cap B}(x)$ (Figura 3.6) é definida ponto a ponto para todo $x \in U$ por:

$$\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x) \quad \text{ou} \quad \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

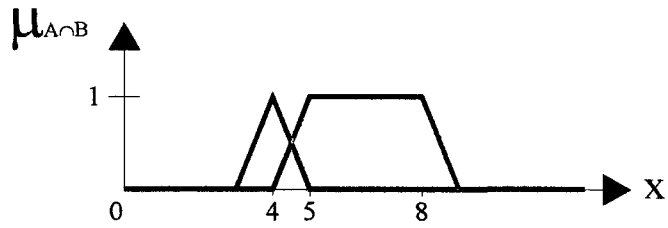

Figura 3.6 Operação do conjunto intersecção.

- Complemento: A função de pertinência $\mu_{\neg A}$ (Figura 3.7) é definida ponto a ponto para todo $x \in U$ por:

$$\mu_{\neg A}(x) = 1 - \mu_A(x)$$

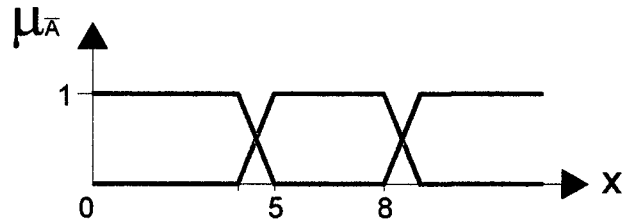

Figura 3.7 Operação do conjunto complemento.

Na teoria de conjuntos nebulosos, a intersecção é implementada por uma família de operadores denominados de **T-Normas** (normas triangulares); a união é implementada por uma família de operadores denominados de **T-Conormas** ou **S-Normas** (conormas triangulares), conforme definidas a seguir (DE AZEVEDO, 2000).

- Normas triangulares (t-normas): As normas triangulares foram introduzidas para modelar distância no espaço métrico probabilístico. Essas normas são exaustivamente usadas para modelar a conectiva *AND*. A Tabela 3.2 mostra as t-normas básicas. A maior norma triangular é a intersecção e a menor é o produto

drástico. As operações associadas à norma triangular são definidas para *todo* $x,y \in [0,1]$. Essas normas são definidas como:

$$T: [0,1]\, x\, [0,1] \rightarrow [0,1]$$

Tabela 3.2 Normas triangulares básicas – t-normas.

Intersecção ou Mínimo	$MIN(x,y) = x \wedge y = \min\{x,y\}$
Lukasiewicz ou Produto Limitado	$LAND(x,y) = x \circ y = \max\{0, x+y-1\}$
Produto Algébrico ou Probabilístico	$PAND(x,y) = x.y = x.y$
Fraca	$WEAK(x,y) = x \wedge y$, se $x \vee y = 1$ 0, caso contrário
Hamacher	$H\gamma(x,y) = x.y/(\gamma + (1-\gamma)(x+y-x.y))$
Dubois and Prade	$sa = x.y/(\max\{x,y,a\}), a \in [0,1]$
Produto Drástico	$x \cap y = x$, se $y = 1$ $x \cap y = y$, se $x = 1$ $x \cap y = 0$, se $x, y < 1$

- Conormas triangulares (t-conormas): As conormas triangulares são usadas para modelar a conectiva *OR*. A Tabela 3.3 mostra as t-conormas básicas. As t-conormas são definidas pelo seguinte mapeamento:

$$S: [0,1]\, x\, [0,1] \rightarrow [0,1]$$

Tabela 3.3 Conormas triangulares básicas – t-conormas.

União ou Máximo	$MAX(x,y) = x \vee y = \max\{x,y\}$
Lukasiewicz ou Soma Limitada	$LOR(x,y) = x \oplus y = \min\{1, x+y\}$
Soma Algébrica ou Soma Probabilística	$POR(xy) = x + y - x.y$
Forte	$STRONG(x,y) = x \vee y$, se $x \wedge y = 1$ 1, caso contrário
Soma Disjunta	$x \Delta y = \max\{\min(x, 1-y), \min(1-x, y)\}$
Soma Drástica	$x \cup y = x$, se $y = 0$ $x \cup y = y$, se $x = 0$ $x \cup y = 0$, se $x, y > 0$

Um outro operador extremamente importante é aquele da implicação *fuzzy* (SANDRI, 1999). A implicação *fuzzy* (ou regras *fuzzy* IF-THEN, ou condição de estado *fuzzy* ou simplesmente regra *fuzzy*) assume a seguinte forma:

$$\text{SE } x \text{ é } A \text{ ENTÃO } y \text{ é } B \text{ ou } A \rightarrow B$$

onde, A e B são valores lingüísticos definidos por conjuntos *fuzzy* no universo de discurso X e Y, respectivamente. A implicação *fuzzy* divide-se em duas partes, uma chamada de antecedente ou premissa (x é A) e outra denominada conseqüente ou conclusão (y é B). Exemplo de regra *fuzzy*:

SE pressão é alta ENTÃO volume é pequeno

A regra *fuzzy* pode ser definida como uma relação binária *fuzzy* R no produto do espaço X e Y, uma vez que descreve a relação entre duas variáveis x e y. Portanto, $A \rightarrow B$ deve ser definida ponto a ponto, e isso é feito através dos operadores de implicação *fuzzy* (Tabela 3.4).

Tabela 3.4 Operadores de implicação *fuzzy*.

Larsen	$x \rightarrow y = xy$
Lukasiewicz	$x \rightarrow y = \min\{1, 1 - x + y\}$
Mamdani	$x \rightarrow y = \min\{x, y\}$
Standart Strict	$x \rightarrow y = 1$, se $x \leq y$ 0, caso contrário
Gödel	$x \rightarrow y = 1$, se $x \leq y$ y, caso contrário
Gaines	$x \rightarrow y = 1$, se $x \leq y$ y/x, caso contrário
Kleene-Dienes	$x \rightarrow y = \max\{1 - x, y\}$
Kleene-Dienes-Luk	$x \rightarrow y = 1 - x + x.y$

Agora, explica-se o funcionamento de um sistema *fuzzy*. Para explanar o mecanismo do raciocínio nebuloso, suponha-se o seguinte sistema apresentado na Figura 3.8 (BARRETO, 1997). Este sistema mapeia entradas numéricas em saídas numéricas, sendo basicamente constituído por quatro componentes:

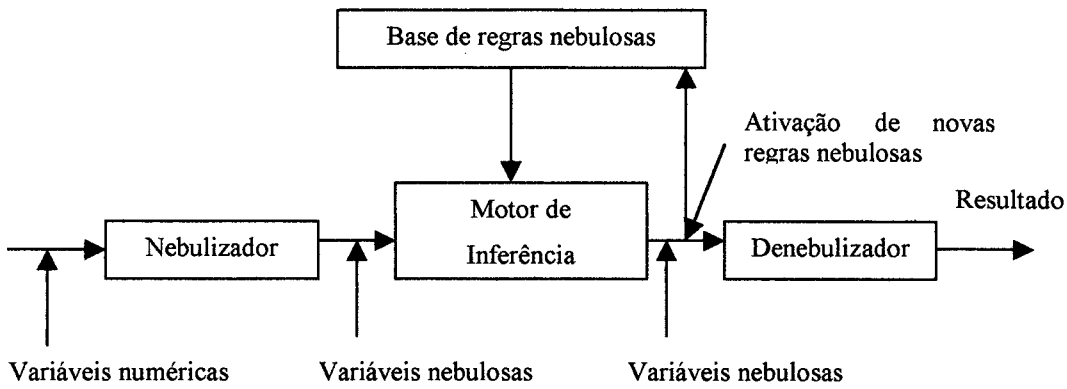

Figura 3.8 Sistema de inferência nebulosa.

- Regras: São regras no formato SE...ENTÃO, normalmente extraídas de dados numéricos ou do especialista.

  Precisa-se de uma compreensão de:

  1. Variáveis lingüísticas versus valores numéricos das variáveis (por exemplo, idade versus 35 anos)
  2. Variáveis lingüísticas quantificadoras (dor pode ter um número finito de termos lingüísticos relacionados com ela, eles podem ir desde extremamente forte a quase nada) o qual é feito utilizando funções de pertinência nebulosa.
  3. Conexões lógicas para variáveis lingüísticas, por exemplo, *e, ou*, etc.
  4. Implicações, por exemplo, Se *A* então *B*.

  E além de tudo isto, precisa-se entender como combinar mais de uma regra.

- Nebulizador: Responsável pelo mapeamento da entrada, que é um valor numérico, numa variável lingüística. Esta ação é indispensável, pois desta forma são ativadas as regras que estão em termos de variáveis lingüísticas, as quais possuem conjuntos nebulosos associados a eles.

- Motor de Inferência: Faz o mapeamento de conjuntos nebulosos em conjuntos nebulosos, ou seja, manipula o caminho no qual as regras são combinadas, de modo a se obter a distribuição de saída *fuzzy*. Geralmente é utilizado o operador de união.

- Denebulizadores: Em muitos casos é desejável que a saída do sistema não seja *fuzzy* e sim uma saída definida. A etapa de *desfuzzificação* mapeia a saída *fuzzy* em um valor definido. Isto é necessário porque um número pode corresponder a ações diferentes dependendo da aplicação em que está sendo usado. Por exemplo, em uma aplicação de processamento de sinais, um número poderia corresponder a uma predição do comportamento do sinal observado, e em uma aplicação de diagnóstico médico, a um prognóstico da doença.

  Muitos denebulizadores foram propostos na literatura, embora não existam bases científicas para nenhum deles. Em conseqüência, denebulizar é uma arte mais do que uma ciência (BARRETO 1997). Algumas estratégias de *desfuzzificação* são:

  1. **Método do máximo critério (MAX):** O máximo critério acha o ponto no qual a distribuição de saída possui o seu primeiro valor de máximo (Figura 3.9).
  2. **Método da média dos máximos (MOM):** O MOM acha o valor médio onde a saída possui seus máximos valores. Isto pode ser computado como (Figura 3.9):

     $$zo = \sum (w_j / l)$$

     onde, w_j são os valores no qual a função de pertinência possui seus máximos valores e l é o número de valores máximos encontrados.
  3. **Método de centro de massa (COA):** É largamente usado. Essa estratégia procura pelo centro de gravidade da distribuição de saída *fuzzy* (Figura 3.9).

O sistema lógico clássico é poderoso para resolver problemas especificados em termos de dois valores (binário). O sistema lógico *fuzzy*, como extensão dos sistemas clássicos, consegue manipular problemas em que há vários estados de decisão, incluindo o binário. Além disso, o sistema *fuzzy* trata com conceitos lingüísticos, como *mais*, *menos*, *maior*, de uma forma parecida com a do ser humano.

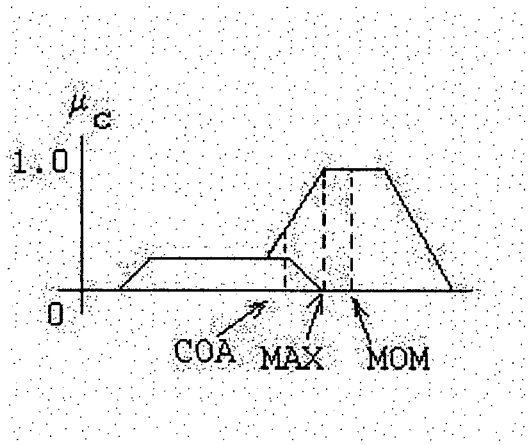

Figura 3.9 Estratégias de desfuzificação.

O procedimento de raciocínio *fuzzy* é paralelo, ou seja, a combinação da parte antecedente da regra com a entrada pode ser computada de forma paralela. Isso faz com que o procedimento *fuzzy* seja apropriado para ser implementado em processadores paralelos.

Pesquisa e desenvolvimento estão em andamento em aplicações *fuzzy* em projetos de software, incluindo sistemas especialistas *fuzzy* e integração de lógica *fuzzy* com redes neurais, os denominados algoritmos genéticos adaptativos, com o objetivo de construção de um sistema *fuzzy* capaz de aprender (ADLASSNIG, 1986).

A teoria de conjuntos *fuzzy* apresenta propriedades que a tornam interessante para formalizar a informação incerta sobre a qual o diagnóstico e tratamento médico estão baseados. Primeiramente, porque define entidades médicas inexatas como conjuntos *fuzzy*. E em segundo lugar, ela fornece uma abordagem lingüística com uma excelente aproximação com textos. Finalmente, a lógica *fuzzy* oferece métodos de raciocínio capazes de realizar inferências aproximadas. Estes fatos sugerem que a teoria de conjuntos *fuzzy* oferece uma base razoável para desenvolvimento de um sistema de diagnóstico computadorizado (ADLASSNIG, 1986).

4. *PROLOG* E RESOLUÇAO DE PROBLEMAS

Como mencionado anteriormente, a linguagem utilizada para a construção da máquina de inferência do *shell* que será apresentado é *Prolog*. Desta forma, torna-se necessário dar uma introdução às principais características desta linguagem que vem sendo amplamente utilizada em aplicações de IA e explicar o seu mecanismo de funcionamento, ou seja, o princípio da resolução. Também, dá-se uma pequena introdução ao raciocínio médico, baseado no mecanismo conhecido como *abduction*.

4.1 Prolog

Prolog significa *Programming Logic*, ou Programação em Lógica. Foi criado por volta de 1970, por Alan Colmerauer e seus colegas na Universidade de Marseille (ROBINSON, 1988). O seu objetivo era criar uma linguagem que fosse adequada para problemas relacionados à análise de linguagem natural e à sua compreensão (BARRETO, 1997). *Prolog* tornou-se, rapidamente, o líder das linguagens de IA na Europa, enquanto LISP, que é uma outra linguagem utilizada para resolver este mesmo tipo de problemas, era utilizada nos Estados Unidos (ROBINSON, 1988).

O interesse pelo *Prolog* realmente cresceu após os japoneses lançarem o seu projeto da Quinta Geração de Computadores.

Prolog é uma linguagem declarativa, ou seja, consiste de declarações ou descrições sobre uma interpretação, isto é, quais as hipóteses que são verdadeiras em uma interpretação, diferenciando de linguagens como C que são linguagens procedurais, onde é necessário definir todos os passos a serem executados pelo programa.

O conjunto de declarações é também chamado de base de dados para *Prolog*. Para determinar se uma dada pergunta feita pelo usuário é ou não verdadeira para a interpretação, *Prolog* usa sua base de dados e aplica suas regras de inferência, utilizando-se da lógica dos predicados e do princípio da resolução para resolver um determinado problema.

A base de dados é composta de fatos e regras. A seguir, explicam-se os conceitos de fatos e regras e outros conceitos importantes para a construção de um

programa *Prolog*. Deve-se considerar o fato do Prolog ter diferentes versões, que apresentam diferenças de sintaxe. A sintaxe levada em consideração nos exemplos apresentados é a sintaxe utilizada pelo *Visual Prolog* 5.2, que foi a versão utilizada para a implementação do *shell* proposto.

- **Fatos**

Os fatos são relações ou propriedades que o programador sabe que são verdadeiras (CLOCKSIN, 1987).

"Jorge é pai de Pedro" é um fato, onde Jorge e Pedro são os objetos e *ehPai* é o relacionamento. A sintaxe para escrever um fato é da seguinte forma:

ehPai (jorge,pedro).

ou seja, a relação seguida dos objetos, que devem ser iniciados com letra minúscula e separados por vírgulas, entre parênteses, seguido do ponto final.

A ordem dos objetos é fundamental. Ela deve ser consistente em toda a programação. Neste caso, a ordem é *(pai, filho)*.

Os objetos são chamados de argumentos. E a relação é chamada de predicado. Então, pode-se dizer que *ehPai* é um predicado que tem dois argumentos.

Quando temos uma coleção de fatos temos uma base de dados (*DATABASE*).

- **Questões**

Após se montar uma base de dados com os fatos a respeito de um determinado problema, pode-se fazer questionamentos ao programa *Prolog*. Ao ser questionado, o programa *Prolog* irá percorrer a base de dados à procura de um fato que combine com o fato perguntado. A base é percorrida sempre do início para o final. Um fato combina com a questão feita se os predicados são iguais e os argumentos também, inclusive em número.

Vamos supor que a base de dados contenha os seguintes fatos:

ehPai (jorge,pedro).

ehPai (jorge,josé).
ehPai(pedro,alfredo).

Ao se consultar o programa, perguntado se:

?- ehPai (jorge,pedro).

recebe-se como resposta *yes*. Porém, se a pergunta for

?- ehPai (jorge,joao).

recebe-se como resposta *no*.

A resposta *no* não significa necessariamente que o fato é falso, mas sim, que de acordo com os dados disponíveis, a resposta é provavelmente negativa. Na vida real, Jorge poderia ser também pai de João, só que este fato pode não estar declarado na base de dados (CLOCKSIN, 1987).

Uma questão feita ao *Prolog* é comumente chamada de *Goal* (objetivo).

➢ **Variáveis**

Em *Prolog*, tem-se também o conceito de variável, ou seja, quando se quer todos os valores possíveis para um determinado argumento, substitui-se o mesmo por uma variável. Como exemplo, se é necessário saber todos os filhos de Jorge, substitui-se o valor do argumento correspondente aos filhos por uma variável *Filhos*. As variáveis devem iniciar com letra maiúscula.

Em *Prolog*, tem-se o seguinte:

?- ehPai (jorge,Filhos)
Filhos=pedro
Filhos=josé

Prolog pesquisa a base de dados, procurando todos os predicados *ehPai* onde o primeiro argumento é *jorge*. Inicialmente, a variável *Filhos* está não-instanciada.

Quando isto acontece, *Prolog* permite que esta variável combine com todos os argumentos na mesma posição no fato, devolvendo-os como resposta.

Existe também a possibilidade da utilização de variáveis anônimas, que é representada pelo sinal de sublinhado (_) (CLOCKSIN, 1987). Utiliza-se este recurso quando um argumento de um predicado pode ser ignorado. Por exemplo, se quisermos saber apenas as pessoas que são pais, mas não nos interessa saber quem são os filhos. Então, basta fazer ao *Prolog* a seguinte questão:

?- ehPai (Pai,_).

Com isto, o programa *Prolog* retorna todos os valores possíveis para *Pai* independente dos valores do segundo argumento. E como resposta obteremos:

Pai=jorge
Pai=pedro

- **Conjunções e Disjunções**

Quando se deseja combinar fatos e verificar se os dois são satisfeitos, pode-se utilizar a conjunção *and* para ligá-los. Esta conjunção é representada pela vírgula (,). Por exemplo, quando se quer saber se Pedro e José são filhos de Jorge, a seguinte pergunta deve ser feita ao *Prolog*:

?- ehPai (jorge,jose), ehPai(jorge,pedro)

e como resposta, obtém-se *yes*.

O *Prolog* irá tentar satisfazer cada um dos objetivos da pergunta na ordem em que elas se apresentam, percorrendo a base de dados. Ou seja, primeiro o *Prolog* vai tentar encontrar um fato na base de dados que satisfaça *ehPai (jorge,jose)*. Se este objetivo for satisfeito, ele tentará satisfazer o segundo, *ehPai (jorge,pedro)*. Se este também for satisfeito, a resposta é positiva, caso contrário, é negativa. Pode ser utilizado qualquer número de fatos na questão.

O uso de conjunções pode ser combinado ao uso de variáveis para se fazer

questões mais elaboradas. Pode-se, por exemplo, querer saber quem é o pai de Pedro e José. A pergunta a ser feita pelo *Prolog* é a seguinte:

?- ehPai(X,pedro), ehPai(X,jose)

E como resposta obtém-se:

X=jorge.

Prolog tenta satisfazer o primeiro fato. Ao encontrar na base de dados o fato *ehPai (jorge,pedro)*, ele instancia a variável *X* com o argumento *jorge* e então, passa a tentar satisfazer o segundo fato procurando por *ehPai (jorge,jose)*. Se o segundo fato é satisfeito, ele devolve o valor com que *X* está instanciado (no caso, *jorge*) como resposta.

Pode também se utilizar uma disjunção, ou um *or*, que é representado pelo ponto e vírgula (;). Se qualquer um dos objetivos for satisfeito, então a regra é satisfeita.

- **Regras**

Usa-se uma regra num programa *Prolog*, quando um fato depende de um grupo de outros fatos. A regra tem a forma Se...Então e é composta de uma cabeça e de um corpo, conectados pelo símbolo (:-) que é composto de dois pontos e hífen e pode ser lido como *se*. A cabeça descreve qual fato a regra pretende definir e o corpo descreve qual o conjunto de fatos que deve ser satisfeito para que a regra seja verdade. Como exemplo, vamos supor que ao programa *Prolog* seja adicionado um predicado que representa a relação de avô, o predicado *ehAvo*. Sabe-se que: *X* é avô de *Z* se *X* é pai de *Y* e *Y* é pai de *Z*. Então, obtém-se a seguinte regra escrita na forma *Prolog*:

```
ehAvo (X,Z) :-
        ehPai (X,Y),
        ehPai (Y,Z).
```

Após adicionar esta regra ao programa *Prolog*, pode-se fazer a seguinte pergunta:

?- ehAvo (jorge,alfredo).

Considerando-se a regra, a variável *X* foi instanciada para *jorge* e a variável *Z* foi instanciada para *alfredo*. Então, o *Prolog* irá tentar satisfazer o primeiro predicado do corpo da regra que se torna *ehPai (jorge, Y)*. O *Prolog*, irá tentar combinar o fato obtido com algum fato da base de dados. E vai encontrar *ehPai(jorge,pedro)*, instanciando *Y* para *pedro*. Estando o primeiro fato satisfeito, *Prolog* passará a tentar resolver o segundo fato *ehPai (pedro, alfredo)* e irá percorrer novamente a base de dados, encontrando o fato. Assim, a resposta dada pelo *Prolog* será *yes*.

Um programa *Prolog* será sempre definido em função de fatos e regras que são as cláusulas do programa (CLOCKSIN, 1987).

➢ **Seções de um programa**

Um programa está organizado em seções, onde são declaradas os predicados e argumentos, as regras e objetivos. As 4 principais seções de um programa são (CLOCKSIN, 1987):

- Seção *Clauses*: Nesta seção, são colocados os fatos e regras que o programa opera enquanto tenta satisfazer um objetivo;
- Seção *Predicates*: é onde se declaram os predicados e os domínios (ou tipos) dos argumentos deste predicado;
- Seção *Domains*: onde se declaram os domínios que não são padrão da linguagem *Prolog*;
- Seção *Goal*: onde se declara o objetivo inicial do programa. Um objetivo interno ao programa, permite que ele execute independente do ambiente de desenvolvimento.

➢ **Unificação e *Backtracking***

São dois mecanismos utilizados pelo *Prolog*, quando está à procura de uma resposta para um objetivo dado.

Prolog usa unificação para tratar com problemas com variáveis. Unificação é o

processo de tornar predicados iguais pela substituição de variáveis por argumentos, e transportar estas substituições para a regra como um todo (ROBINSON, 1988).

Por exemplo, quando *Prolog* tenta satisfazer o seguinte predicado:

?- ehPai(X,alfredo).

ele procura pela base de dados (do início para o final) um fato que se encaixe com a pergunta feita e ao encontrar *ehPai(pedro,alfredo)*, ele combina a variável *X* com *pedro* e as cláusulas ficam idênticas, ou seja, diz-se que o objetivo unificou com o fato.

Quando o *Prolog* falha na tentativa da busca de uma solução, ele procura um outro caminho para tentar satisfazer o objetivo. Esta método é chamado *backtracking.* Por exemplo, quando o *Prolog* está procurando por uma solução, ele pode possivelmente ter que decidir entre dois casos possíveis. Então, ele deve marcar com um *backtracking point* o fato que ele vai testar. Se este fato falha, ele retorna ao *backtracking point* e passa a testar o próximo fato (ROBINSON, 1988).

➢ **Recursão**

O antecedente de uma regra pode depender da mesma regra, caso no qual a regra é definida em função dela mesma. Uma função em que o item sendo definido é, ele próprio, parte da definição é chamada de função recursiva.

➢ ***Cut, Fail***

Existem predicados pré-definidos que implementam funções úteis no desenvolvimento de um programa. Vamos apresentar dois deles: um para quando se quer evitar o *backtracking,* e outro para quando se quer forçá-lo.

Cut (!) diz ao programa *Prolog* que não há necessidade de considerar novamente escolhas prévias, ou seja, a partir do momento que o programa processou um predicado *cut*, não é possível voltar (*backtrack*) a objetivos anteriores a ele.

Ele é importante em dois casos:

- O *Prolog* não perde tempo tentando satisfazer objetivos que se pode dizer de antemão que não irão contribuir para a solução do problema;
- Menor ocupação de memória, porque os pontos de backtracking não precisam ficar registrados para uso posterior.

fail é um predicado que sempre retorna falso. Pode ser útil no caso de se querer forçar um backtracking de modo a achar outras soluções possíveis.

➢ **Base de Dados**

Uma característica do *Prolog* que foi de fundamental importância para o desenvolvimento deste sistema, permitindo que ele seja genérico, é o caso da base de dados poder ser lida a partir de um arquivo em tempo de execução do programa a partir do predicado pré-definido *consult*, e da possibilidade de se incluir e remover predicados através dos predicados *retract* e *assert*.

O predicado *consult* tem um parâmetro que é o nome do arquivo a partir do qual serão lidos os fatos a serem adicionados à base de dados, ou seja, sua sintaxe é:

consult(<nome do arquivo>).

As frases que são lidas no arquivo são inseridas em uma espécie de biblioteca, onde podem ser acessadas sempre que forem necessárias para responder a uma pergunta (ARARIBÓIA, 1989).

O predicado *assert* permite introduzir fatos na base de dados sem se precisar do comando *consult*. Permite a introdução de apenas um fato a base (ARARIBÓIA, 1989). A sua sintaxe é:

assert(<o fato>).

O predicado *assert* apresenta variações:

- *asserta(<o fato>)*: Insere um fato antes dos fatos existentes na base de dados com o mesmo predicado;

- *assertz(<o fato>)*: Insere um fato depois dos fatos existentes na base de dados com o mesmo predicado. a*ssert* comporta-se como *assertz.*

O predicado *retract* remove um fato da base de dados. Sua sintaxe é:

retract(<o fato>).

Existe o predicado *retractall* que remove todos os fatos que apresentam o predicado indicado como parâmetro e sua sintaxe é:

retractall(<o fato>).

Os predicados que são inseridos em tempo de execução já devem estar declarados na seção *predicate*, devendo-se levar em consideração que apenas fatos podem ser inseridos.

4.2 Resolução de Problemas

Uma coisa óbvia a investigar quando se tem um conjunto de proposições é se alguma coisa pode ser obtida a partir deste conjunto. Ou seja, quais as conseqüências que as proposições têm. Então, aquelas proposições que se está considerando como verdadeiras são os axiomas ou hipóteses e aquelas proposições que são obtidas a partir destas são os teoremas (GERSTING, 1995).

A atividade de derivar (ou provar) conseqüências a partir das proposições dadas é chamada de Prova de Teoremas ou Resolução de Problemas. Estas provas são obtidas através da aplicação das regras de inferência para uma dada Lógica, onde estas regras de inferência dizem o que se pode concluir de uma ou mais proposições. Por exemplo, no cálculo proposicional, a regra *modus ponens*, onde se B é conseqüência de A e A é verdade, então B é verdade. Formalmente, tem-se: $A \rightarrow B, A \Rightarrow B$.

A regra de inferência usada pelo *Prolog* é chamada de Resolução. O Princípio da Resolução foi descoberto por ROBINSON (1965). Usando-se a resolução, pode-se provar teoremas de uma maneira puramente mecânica a partir de nossos axiomas. Tem-

se somente que decidir a quais proposições aplicá-la e conclusões válidas serão produzidas automaticamente.

A regra de inferência da resolução é projetada para trabalhar com as fórmulas na forma clausal conhecida como cláusulas de Horn (GERSTING, 1995).

Os fatos e regras em uma base de dados podem ser definidos em termos de "fórmulas bem formadas" (wffs). E os fatos e regras são exemplos das cláusulas de Horn, que é uma wff composta por predicados ou pelas negações dos predicados (com variáveis ou constantes como argumentos) unidos por disjunções, onde no mínimo um predicado não é negado (GERSTING, 1995).

A resolução é um processo de eliminação definido pela inferência, chamada fórmula da resolução:

$$((\neg A \vee B) \wedge A) \rightarrow B$$

cujo membro esquerdo, por distribuição de $\wedge$, resulta em $(\neg A \wedge A) \vee (B \wedge A)$, que em virtude da falsidade da contradição $\neg A \wedge A$ resulta em $B \wedge A$, o que implica que B é verdade. Também são importantes as equivalências:

$$\neg(A \rightarrow B) \Leftrightarrow \neg(\neg A \vee B) \Leftrightarrow A \wedge \neg B$$

Suponha a proposição $A \rightarrow B$, onde A é um conjunto de premissas (um mundo). *Prolog* tenta demonstrar sua negativa $\neg(A \rightarrow B)$, que é equivalente a $A \wedge \neg B$. Isto implica provar que B é falso, presumindo que A é verdade (isto é, provar que $\neg B$ pode existir no mundo A). Se esta prova falha, se supõe que a proposição é válida. Para fazer isto, *Prolog* adiciona a A a proposição $\neg B$ e por resolução e união procura por uma declaração vazia. Isto é obtido somente em dois casos: primeiro, se B não é válido, segundo, se o conjunto de regras A não é completo. Desta maneira, todo conjunto de regras e toda hipótese B resulta em uma solução. Se a solução é que B é verdade, então B é verdade. Por outro lado, se a solução é que B é falso, isto significa que somente o conjunto de regras não é suficiente para provar que B é verdade (ALGARVE, 1995).

Além disso, *Prolog* usa unificação para tratar com problemas com variáveis (cálculo dos predicados). Unificação é o processo de tornar predicados iguais pela substituição de variáveis por argumentos, e transportar estas substituições para a regra

como um todo. Por isto, cálculo dos predicados é transformado em cálculo das proposições, ao qual a resolução é aplicada (ALGARVE, 1995).

4.3 Raciocínio Inferencial

Este tipo de raciocínio é baseado em regras válidas de inferência, ou seja, se baseia no uso dos silogismos enunciados por Aristóteles. No caso apresentado, a regra de inferência a ser utilizada é a da resolução conforme visto na seção anterior. No entanto, é sabido que especialistas humanos nem sempre se utilizam de regras de inferência válidas. Por exemplo, no caso de especialista médicos, acredita-se que a regra de inferência usada é a da abdução[1] ou raciocínio abdutivo (BARRETO, 1997, BRASIL, 1999). O problema deste mecanismo é que não se trata de uma inferência válida, ou seja, não se apóia em nenhum dos silogismos.

Formalmente, a abdução pode ser representada por: $A \rightarrow B, B \Rightarrow A$. Ou seja, se B é conseqüência de A e B é verdade, conclui-se que A é verdade (BRASIL, 1999). No entanto, B poderia ser conseqüência de C sendo verdade sem que A fosse verdade. Na verdade, abdução poderia quando muito indicar uma probabilidade, nunca uma inferência (BARRETO, 1997). A abdução seria como um *modus ponens* (visto anteriormente) invertido.

Para o caso médico, tem-se que se uma doença A implica num sintoma B, um paciente apresentando um sintoma B é suspeito de estar com a doença A. Mas este sintoma poderia ser causado por mais de uma doença. A abdução significaria, então, uma evidência plausível.

O raciocínio médico é ainda pouco conhecido, sendo difícil simulá-lo em um computador. O médico tem dificuldade de articular de maneira lógica, direta e consistente como ele chegou a uma determinada conclusão. Ele envolve no seu raciocínio, simultaneamente, processos lógicos, avaliação probabilística, encadeamento causal entre outros. Este fato representa dificuldade, principalmente para os processos de elicitação e representação do conhecimento.

O raciocínio médico, com o propósito de obter um diagnóstico, é composto por

---

[1] Como não existe tradução para o termo *abduction*, neste trabalho, utiliza-se o termo abdução.

alguns elementos básicos, como mostrado na Figura 4.1.

Num primeiro passo, o médico faz inúmeras observações e realiza inferências a partir destas. Estas observações incluem história do paciente, as principais queixas, testes laboratoriais, os conhecimentos a respeito de casos semelhantes e a experiência prévia do médico. Estas inferências são feitas combinando, integrando e interpretando os dados até que se obtenha uma classe de diagnóstico aceitável. O médico utiliza heurísticas para analisar estes dados, baseado em suas antigas experiências.

A partir daí, obtém-se uma ou mais hipóteses de diagnóstico. Estas hipóteses são analisadas, passam por um refinamento, para serem validadas ou não. Este processo é executado até a obtenção de um diagnóstico final.

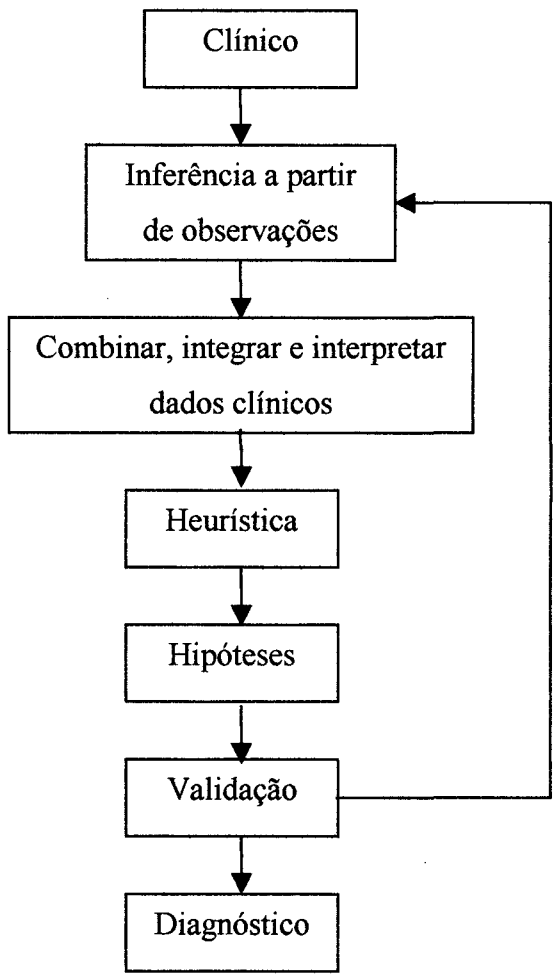

Figura 4.1 Elementos do processo de diagnóstico médico baseado em heurística.

Neste trabalho, há exemplos considerados que são para diagnóstico diferencial

na área médica. Então, poder-se-ia supor que a máquina de inferência deveria ser implementada usando a abdução. No entanto, como explicado anteriormente, esta não é uma regra de inferência válida. Por conseguinte, manteve-se a máquina de inferência baseada na resolução. Os experimentos tem mostrado que utilizando-se de uma base de conhecimento constituída de regras *fuzzy*, o processo de diagnóstico médico pode ser razoavelmente bem simulado por um SE.

5. IMPLEMENTAÇÃO DO SISTEMA

O sistema desenvolvido neste trabalho 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. O sistema implementado recebeu o nome de SSEF (*Shell* para Sistemas Especialistas *Fuzzy*).

A implementação foi dividida em quatro módulos, que implementam as funções necessárias para que o sistema execute as tarefas propostas:

- Criar;
- Modificar;
- Excluir;
- Consultar.

A implementação foi feita utilizando as linguagens de programação Visual *Prolog* 5.2 e *Delphi* 3.0. A linguagem *Borland Delphi* 3.0 é utilizada para implementar os primeiros três módulos e todas as demais funções que se relacionam com a interface do sistema. Esta linguagem, que é uma linguagem visual, foi escolhida por apresentar inúmeras ferramentas que facilitam a implementação de interface. Apresenta, também, funções pré-definidas que facilitam o processo de leitura e escrita em arquivos.

A linguagem *Visual Prolog 5.2 Personal Edition* foi utilizada para implementar a máquina de inferência. A sua escolha foi feita por esta linguagem facilitar o uso do *Prolog*, também fornecendo ferramentas visuais para desenvolvimento de projetos e dlls.

Uma dll (*dynamic link library*) pode ser definida como módulos de código compilado que trabalham em conjunto com um executável para fornecer funcionalidade a uma aplicação, ou seja, uma coleção de rotinas que podem ser chamadas por aplicações e, também, por outras dlls. Um exemplo pode ser uma janela usada como padrão para salvar arquivos, sendo que esta janela é usada por mais aplicações. Tem grande funcionalidade, também, para trabalhar em conjunto com aplicações que estão escritas em uma linguagem diferente da linguagem utilizada para escrever a dll. Por exemplo, pode-se chamar uma dll escrita em *Delphi* de aplicações escritas em *C++*,

Paradox ou *Dbase*, assim como dlls escritas em outras linguagens podem ser chamadas por uma dll escrita em *Delphi*.

A dll é um arquivo compilado separadamente que é linkado em tempo de execução ao programa que a usa. Para se distinguir de executáveis *standalone*, arquivos contendo dlls compiladas são nomeados com extensão .DLL. O programa implementado apresenta um executável principal e duas dlls. A necessidade de criação da dll surgiu, principalmente, do fato da necessidade de se comunicar linguagens diferentes. Isto é, o programa principal, escrito em *Delphi*, quando da realização de uma consulta, precisa chamar uma função escrita em *Prolog*, que será responsável pela ativação da máquina de inferência. A máquina de inferência precisa chamar funções em *Delphi* (já que a proposta do trabalho foi implementar toda a interface 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 executável principal. O esquema pode ser visto na Figura 5.1.

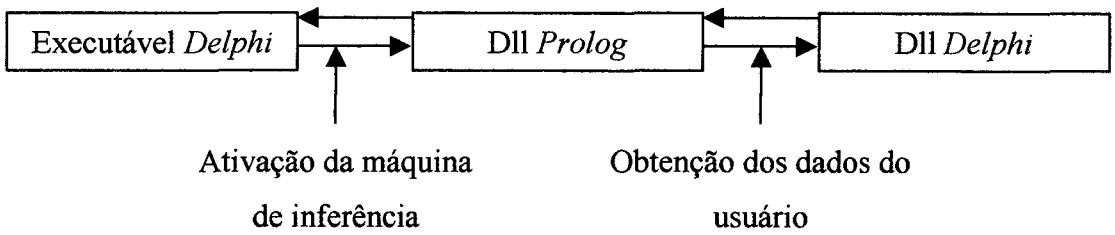

Figura 5.1 Esquema de funcionamento do *shell*.

A partir do executável principal também é possível acionar todos os outros módulos do sistema, como mostrado na Figura 5.2.

O executável principal chama-se *Shell.exe*. A seguir, será mostrado com mais detalhes, o funcionamento de cada módulo e a implementação das dlls.

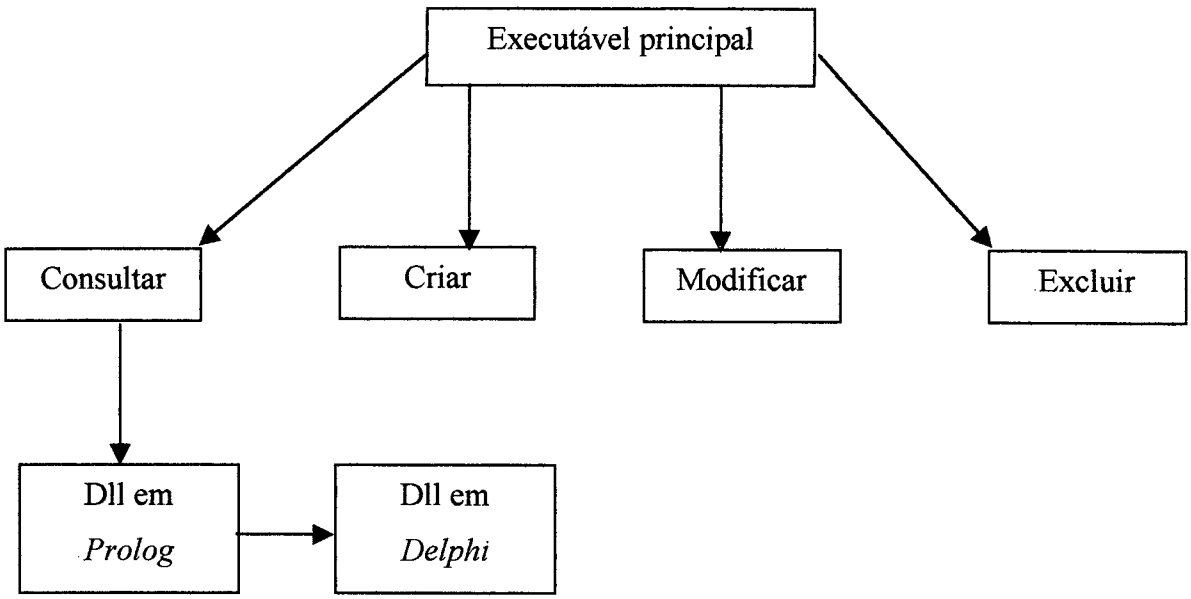

Figura 5.2 Relacionamento entre módulos e dlls do sistema.

5.1 Menu Principal

Antes de detalhar o funcionamento de cada um dos módulos do sistema, será mostrado como está implementada a tela principal, que é responsável pela chamada de todos os módulos.

A Figura 5.3 mostra a tela principal do sistema. Ela apresenta um menu com as seguintes opções:

- *Especialista*: este menu apresenta como sub-opções *Criar*, *Excluir* e *Consultar*, que serão explicados mais detalhadamente ao longo deste capítulo.
- *Gerenciar Base de Conhecimento*: responsável por todo o gerenciamento das informações que estarão disponíveis na base de conhecimento. Também será detalhado posteriormente.
- *Ajuda*: Menu a partir do qual será possível ativar o arquivo de ajuda do sistema.
- *Sair*: Opção de finalização do sistema.

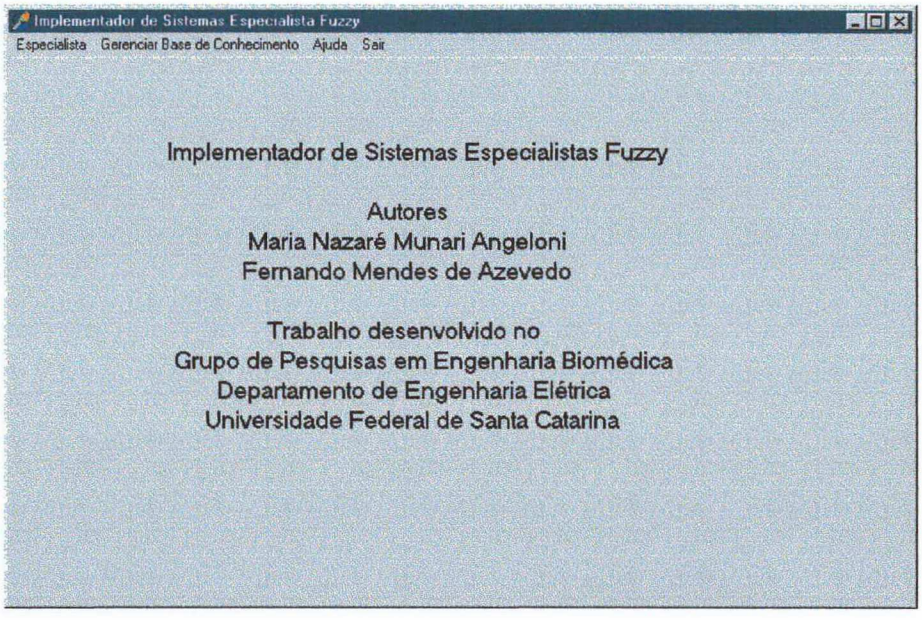

Figura 5.3 Tela principal do sistema.

5.2 Criar

Este módulo permite a criação de um novo SE fisicamente, ou seja, em forma de um arquivo texto, salvo com a extensão .ESP, referente a especialista. Este arquivo texto contém todas as informações que são fornecidas pelo usuário, ou seja, informações gerais do sistema e a base de conhecimento.

Esta opção pode ser acionada a partir do menu *Especialista/Criar* que está na tela principal do sistema, permitindo ao usuário criar um especialista. Para isto, o usuário deve seguir os seguintes passos:

1- Informar uma meta para o especialista. Esta meta é a variável que será o objetivo do sistema;
2- Dar algumas informações gerais sobre o especialista que está sendo criado, como por exemplo, para que serve o especialista e qual o tipo de problema que ele resolve;
3- Pressionar o botão *Salvar* para confirmar a operação. Ao fazer isto, o usuário é solicitado para informar o nome do arquivo sobre o qual deseja salvar o especialista. A extensão, obrigatoriamente, é .ESP.

A Figura 5.4 mostra a tela após a entrada dos dados.

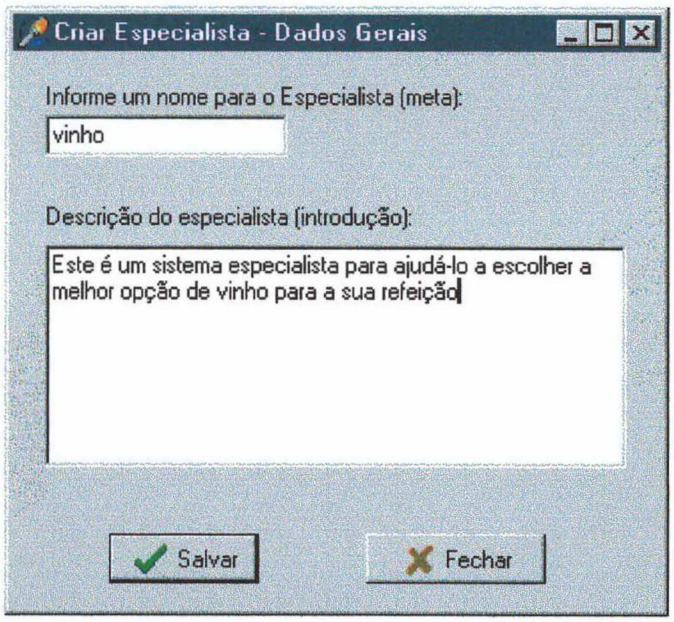

Figura 5.4 Tela para criação do sistema.

5.3 Modificar

Este módulo pode ser acionado pela opção de menu *Gerenciar Base de Conhecimento*. Após criar fisicamente o SE, utiliza-se este módulo para criar e/ou modificar e/ou excluir a introdução, as regras, as perguntas e valores de verdade para uma base de conhecimento. Todas estas operações podem ser feitas utilizando-se a mesma janela. Esta janela apresenta as seguintes pastas *Dados Gerais*, *Valores de Verdade*, *Questões* e *Regras*.

- **Dados Gerais**

Usando esta pasta, o usuário pode modificar a variável meta do sistema e as informações gerais. Para isto, basta pressionar o botão *Alterar*, realizar as alterações desejadas e pressionar o botão *Salvar*.

- **Valores de verdade**

Os valores de verdade serão utilizados na entrada dos dados pelo usuário, quando ele tiver que responder sobre uma pergunta *fuzzy*. A princípio, o sistema permite que se entre com um dos valores que foram pré-determinados na criação da base de conhecimento, ou seja, durante a criação da base de conhecimento, o especialista deve informar grupos ou conjuntos de valores de verdade. Como exemplo, pode-se citar: certeza absoluta (1), quase certo (0,8), não sei (0,5).

A entrada dos valores de verdade é feita utilizando-se da tela mostrada na Figura 5.5.

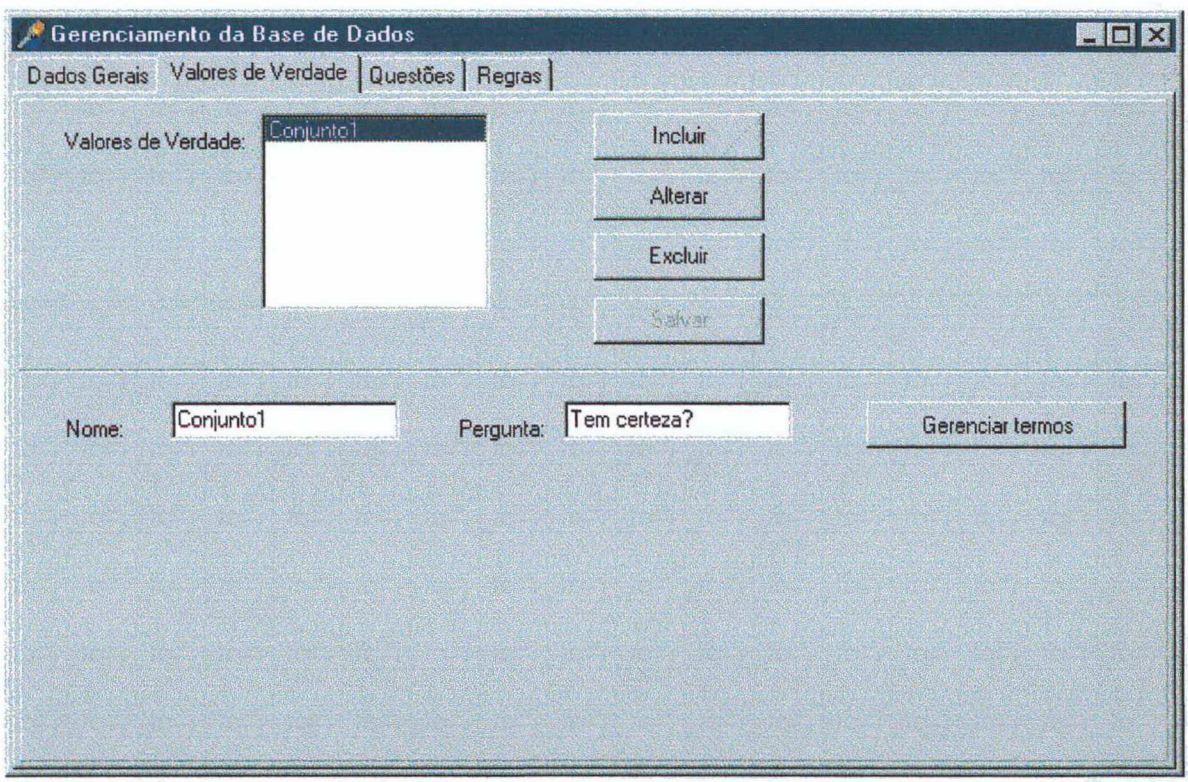

Figura 5.5 Tela para inserção dos valores de verdade.

Para criar um novo grupo de valores de verdade os seguintes passos devem ser executados:

1- Pressionar o botão *Incluir;*
2- Entrar com um nome para identificar um novo conjunto;

3- Entrar com a pergunta que será feita, por exemplo, "Tem certeza?";
4- Pressionar o botão *Gerenciar Termos*, para entrar com os termos do conjunto;
5- Salvar o conjunto, pressionando o botão *Salvar*.

É possível alterar um conjunto de valores de verdade, bastando para isto pressionar o botão *Alterar*, fazendo com que os campos fiquem em modo de alteração. E, também, excluir um conjunto, pressionando o botão *Excluir*.

O gerenciamento dos termos do conjunto é feito pela tela da Figura 5.6.

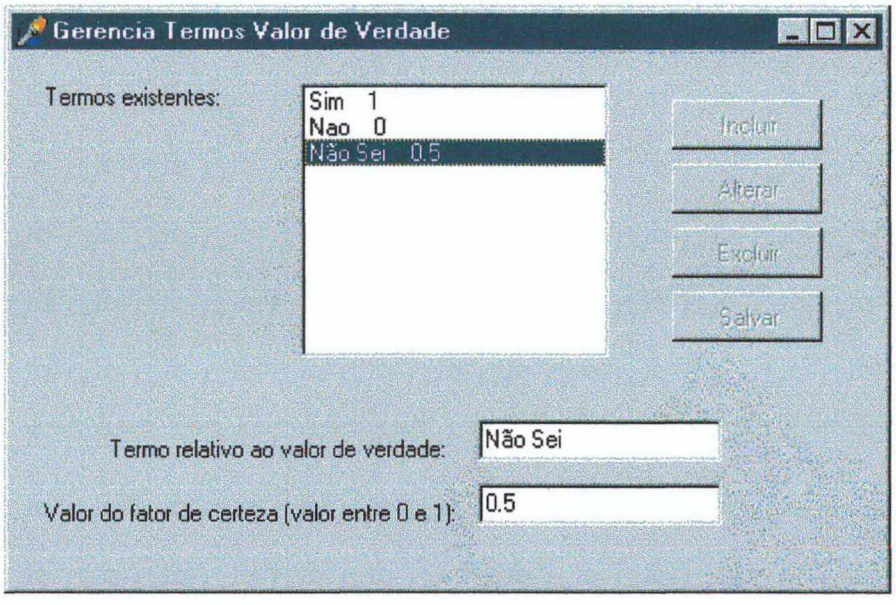

Figura 5.6 Tela para gerenciamento dos termos do conjunto de valores de verdade.

A partir desta tela, é possível entrar com novos termos, alterar e excluir termos existentes. Para entrar com um novo termo, basta:

1- Pressionar o botão *Incluir;*
2- Entrar com um rótulo para o termo, por exemplo, certeza absoluta;
3- Entrar com o valor de verdade, por exemplo, 0,9;
4- Pressionar o botão *Salvar*, para salvar o termo.

É possível alterar um termo, bastando para isto pressionar o botão *Alterar*, fazendo com que os campos fiquem em modo de alteração. E também excluir um termo, pressionando o botão *Excluir*.

➢ Questões

Estas perguntas, representadas por variáveis, serão feitas para o usuário durante a execução de uma consulta, para que seja possível ao SE chegar a uma conclusão.

O gerenciamento de uma questão é feito utilizando-se da tela apresentada na Figura 5.7.

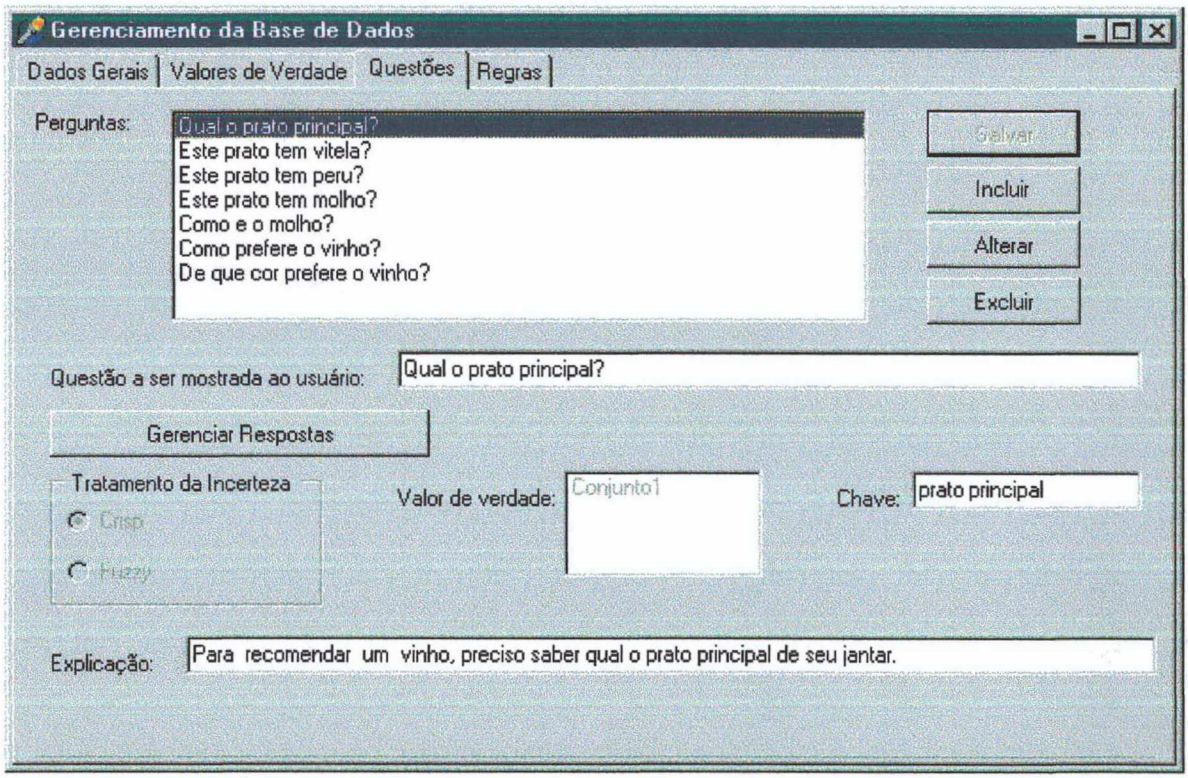

Figura 5.7 Tela para gerenciamento das questões.

Para uma questão, os seguintes dados são necessários: questão que será mostrada ao usuário; tipo da pergunta – *crisp* ou *fuzzy*; se é uma pergunta *crisp*, qual o conjunto de termos de valores de verdade utilizar; informar a chave ou variável que representa a questão; informar as respostas possíveis para a questão; informar uma explicação, que será utilizada para explicar ao usuário o porquê da necessidade daquela pergunta.

Quando se deseja incluir uma nova questão, os seguintes passos devem ser seguidos:

1- Pressionar o botão *Incluir;*
2- Inserir a questão a ser mostrada ao usuário;
3- Escolher o tipo da pergunta;
4- Se for uma pergunta *fuzzy*, escolher qual o fator de certeza a ser utilizado;
5- Entrar com uma chave que será a variável que irá representar a questão;
6- Entrar com a explicação, que será mostrada ao usuário para justificar a necessidade de informar aquela resposta;
7- Pressionar o botão *Gerenciar Respostas* para incluir as respostas possíveis para esta questão;
8- Pressionar o botão *Salvar* para salvar a pergunta.

É possível alterar uma questão, bastando para isto pressionar o botão *Alterar*, fazendo com que os campos fiquem em modo de alteração. E também excluir uma questão, pressionando o botão *Excluir*.

O gerenciamento das respostas possíveis é feito pela tela mostrada na Figura 5.8.

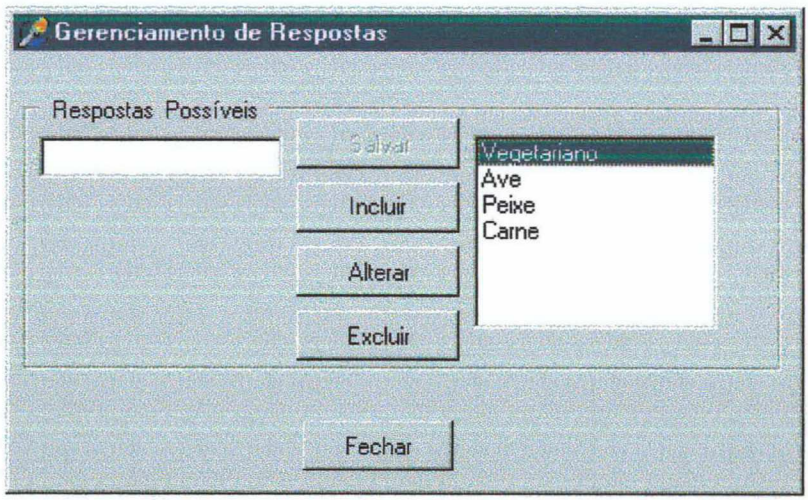

Figura 5.8 Tela para gerenciamento das respostas possíveis.

A partir desta tela, é possível entrar com novas respostas, alterar e excluir respostas existentes. Para entrar com uma nova resposta, basta:

1- Pressionar o botão *Incluir;*

2- Entrar com a resposta possível, por exemplo, Vegetariano;
3- Pressionar o botão *Salvar*, para salvar a resposta.

É possível alterar uma resposta, bastando para isto pressionar o botão *Alterar*, fazendo com que os campos fiquem em modo de alteração. E também excluir uma resposta, pressionando o botão *Excluir*.

As questões representam a variável de interface. É possível expandir o sistema de modo que sejam incluídas também as variáveis intermediárias que não são obtidas do usuário. Para isto, basta criar mais um item no arquivo texto, onde ficariam armazenadas estas variáveis. Seria uma questão de implementação de interface.

➢ **Regras**

As regras serão utilizadas para se chegar à resposta que deve ser dada ao usuário a partir das respostas fornecidas por ele.

As regras têm o formato SE...ENTÃO. Para cada regra, os seguintes dados devem ser fornecidos:

- As premissas da regra. Cada uma delas é composta por:

  "variável" = "resposta possível" é "sim/não"

  Podem ser definidas até 10 premissas.
- A conclusão é composta por:

  "variável" = "reposta possível" com "valor de verdade"

A princípio o sistema aceita apenas uma conclusão, mas pode ser expandido para aceitar mais de uma, evitando que se tenha que repetir um mesmo conjunto de premissas. Isso seria possível modificando-se a interface e realizando-se pequenos ajustes na construção do arquivo texto.

A tela para inserção de regras é mostrada na Figura 5.9.

Para se incluir uma nova regra, os seguintes passos devem ser seguidos:

1- Pressionar o botão *Incluir*;
2- Entrar com a conclusão que é composta por uma variável, uma resposta possível e um valor de verdade;

3- Pressionar o botão *Gerenciar Premissas* para incluir as premissas da regra;
4- Pressionar o botão *Salvar,* para salvar a nova regra.

É possível alterar uma regra, bastando para isto pressionar o botão *Alterar*, fazendo com que os campos fiquem em modo de alteração. E também excluir uma regra, pressionando o botão *Excluir*.

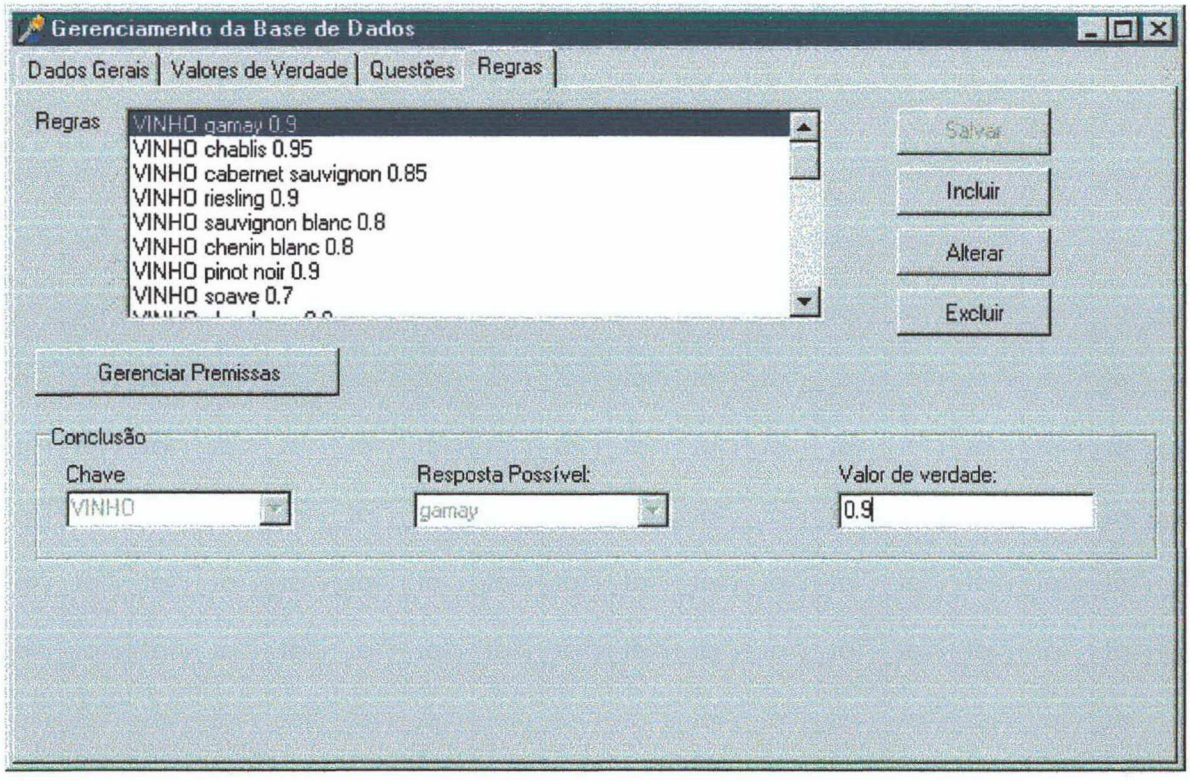

Figura 5.9 Tela para gerenciamento de uma regra.

O gerenciamento das premissas é feito pela tela mostrada na Figura 5.10. Para incluir uma nova premissa, deve-se:

1- Pressionar o botão *Incluir*;
2- Entrar com a premissa que é composta por uma variável, uma resposta possível e uma negação se necessário;
3- Salvar a premissa, pressionando o botão *Salvar*.

É possível alterar uma premissa, bastando para isto pressionar o botão *Alterar*, fazendo com que os campos fiquem em modo de alteração. E também excluir uma premissa, pressionando o botão *Excluir*.

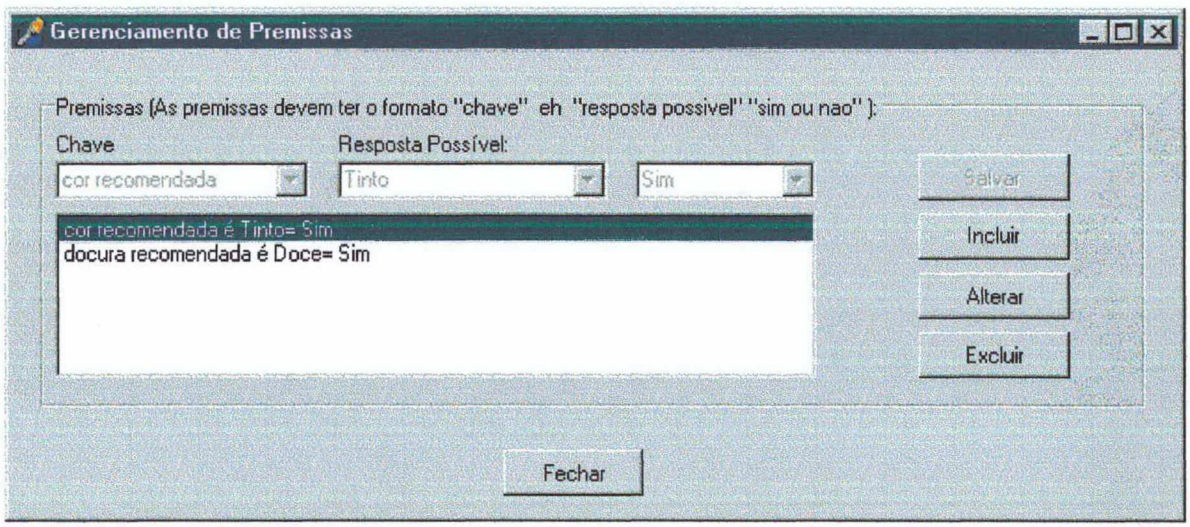

Figura 5.10 Tela para gerenciamento das premissas de uma regra.

5.4 Eliminar

Elimina uma base de conhecimento (ou seja, um "especialista") criado. Para executar esta função, basta selecionar a opção *Especialista/Excluir*.

Para realizar esta tarefa, basta escolher o arquivo .ESP que se deseja excluir. Após pressionar OK, o especialista terá sido removido da memória do seu computador. A Figura 5.11 mostra a tela para exclusão de um especialista.

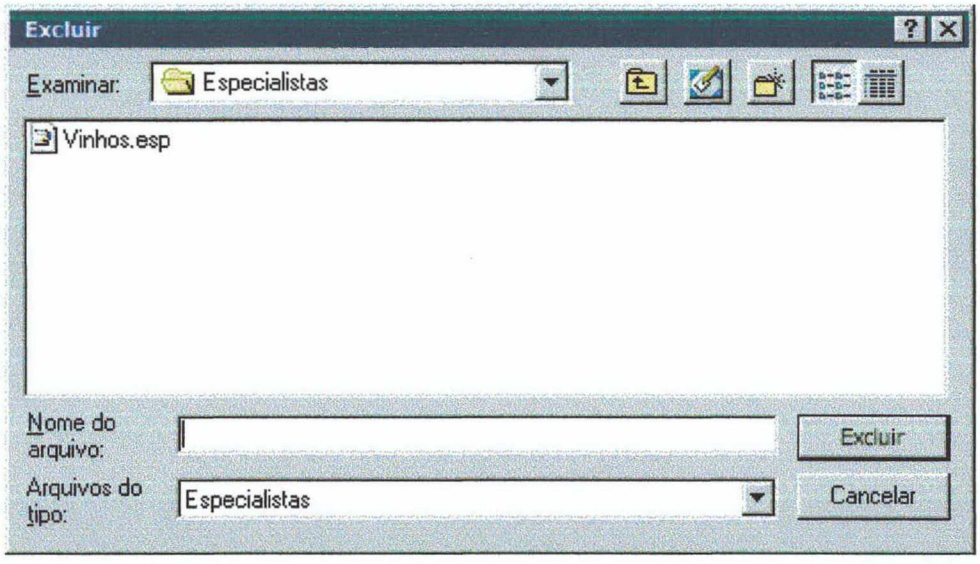

Figura 5.11 Tela para exclusão de um especialista.

5.5 Consultar

O módulo consultar é a "alma" do sistema. Ele é composto por duas dlls. Uma é a máquina de inferência propriamente dita, que é toda implementada em *Visual Prolog*. A outra dll é implementada em *Delphi* e é utilizada para buscar as informações necessárias do usuário. Este módulo pode ser acionado pela opção de menu *Especialista/Consultar*.

Através deste módulo, é possível consultar uma base de conhecimento existente, escolhendo o especialista desejado. Após informar este dado, é gerada, automaticamente, uma seqüência de menus de acordo com as perguntas que foram definidas para este especialista e com as respostas dadas pelo usuário. 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 *Especialista/Consultar* é acionada, pergunta-se ao usuário qual o especialista que ele deseja consultar, permitindo que ele selecione um arquivo .ESP.

Após selecionar o arquivo, é mostrado ao usuário a tela apresentada na Figura 5.12. A partir dela, o usuário pode iniciar a consulta, realizando quantas consultas desejar.

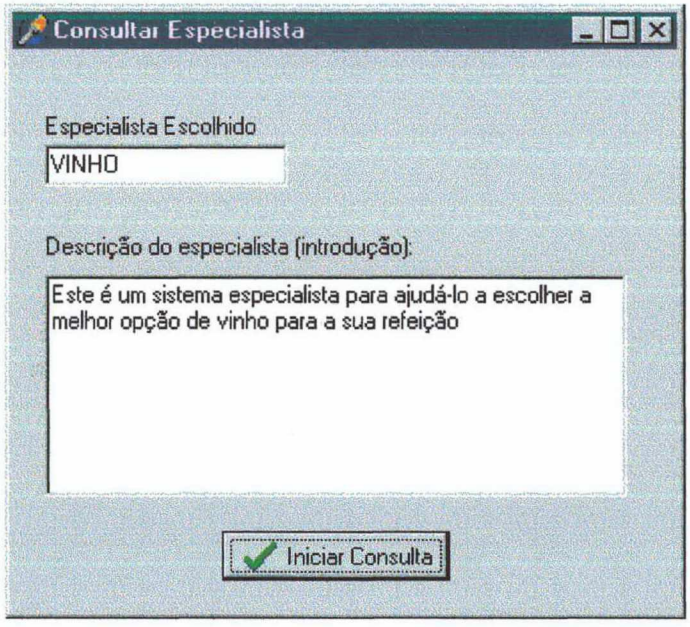

Figura 5.12 Tela de consulta do sistema.

5.5.1 A DLL *Prolog*

A dll implementada em *Prolog*, chamada *MaquinaInferencia.dll*, é a máquina de inferência do sistema. A função pública na dll, que é chamada pelo executável 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, se verifica 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 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-se 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.

Ao voltar a lidar com a regra de partida após calcular todos os valores de verdade para todas as premissas, a máquina de inferência calcula o valor de verdade da regra utilizado-se da seguinte fórmula:

$$\mu(regra) = \mu(conclusão).\min[\mu(premissa1), \mu(premissa2),..., \mu(premissan)] \qquad (5.1)$$

onde μ representa o valor de verdade.

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

A equação 5.1 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 ($A \wedge B = A .B$), que combina o valor de verdade da regra com o valor de verdade obtido a partir das premissas.

Pela fórmula 5.1, a verdade da conclusão sempre varia quando a verdade das regras variam, (fornecendo o mínimo diferente de zero). Por outro lado, a verdade da composição das premissas nunca é maior que a da menor das verdades dos componentes (ALGARVE, 1995).

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 fórmula 5.1.

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).

O programa salva o resultado e fornece uma falha 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 *Prolog* então pesquisa outra solução e assim sucessivamente. Então, quando o *Prolog* testa 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 fuzzy*, que combina os valores de verdade da conclusões obtidas duas a duas, através da fórmula:

$$\mu(final) = \mu(conclusao1) + \mu(conclusao2) - \mu(conclusao1) * \mu(conclusao2) \quad (5.2)$$

onde,

- μ(final) é o valor de verdade final para uma determinada resposta possível;
- μ(conclusão1) e μ(conclusão2) são valores de verdade intermediários obtidos para aquela resposta.

A fórmula 5.2 é obtida a partir da seguinte derivação, considerando-se o $\wedge$ dado pelo produto:

$$A \vee B = \neg(\neg(A \vee B) = \neg(\neg A \wedge \neg B)$$

$$\mu(A \vee B) = \mu[\neg(\neg A \wedge \neg B)]$$

$$\mu(A \vee B) = 1 - \mu(\neg A \wedge \neg B)$$

$$\mu(A \vee B) = 1 - \mu(\neg A).\mu(\neg B)$$

$$\mu(A \vee B) = 1 - [1 - \mu(A)][1 - \mu(B)]$$

$$\mu(A \vee B) = 1 - 1 + \mu(A) + \mu(B) - \mu(A).\mu(B)$$

obtendo finalmente:

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

que é uma equação correspondente à equação apresentada em 5.2.

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 pelo *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 0,9 de valor de verdade, o valor de verdade para a premissa *not corpreferida=tinto* será considerado = 1 - 0,9, ou 0,1.

O arquivo fonte desta dll se encontra no Anexo 1.

5.5.2 A 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 ela recebeu da máquina de inferência. A dll tem como pública duas funções 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.

As Figuras 5.13 e 5.14 mostram as telas que são geradas automaticamente a cada pergunta que deve ser feita ao usuário.

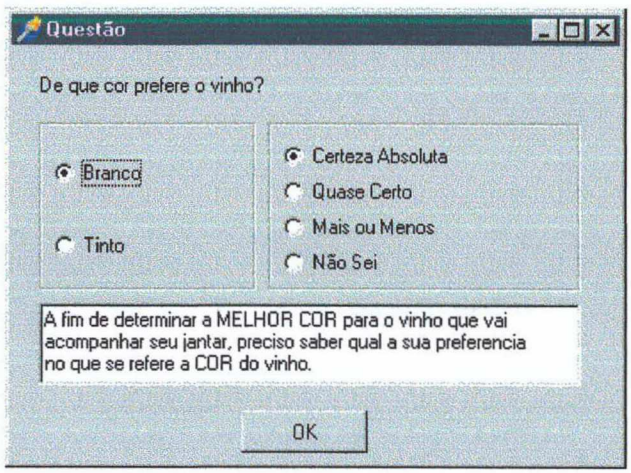

Figura 5.13 Uma pergunta *fuzzy* feita ao usuário.

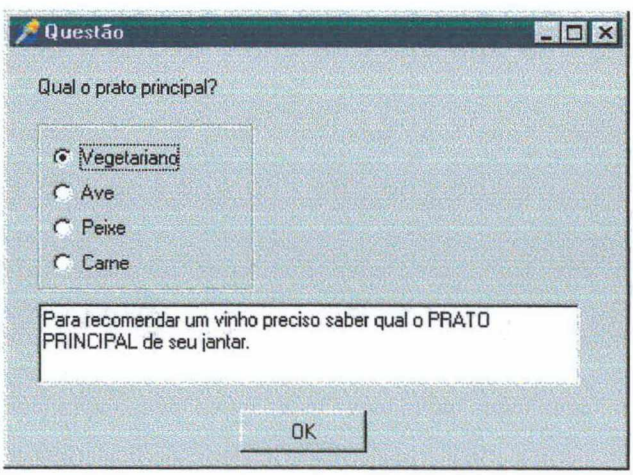

Figura 5.14 Uma pergunta *crisp* feita ao usuário.

5.6 O Arquivo .ESP

O arquivo .ESP, que é a base de conhecimento do SE, criado após as operações de criação e modificação, apresenta-se em forma de predicados, pois esta base é consultada através do predicado *consult* 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.

- 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.

- 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 branco. 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 = sim para condição afirmativa e não para negativa. Faz parte da conclusão;
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 sim para condição positiva ou não para negativa, a variável e a resposta possível.

- 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.

No Anexo 2, encontra-se a base de conhecimento montada para teste para o exemplo dos vinhos, sendo possível observar todos estes predicados.

6. EXPERIMENTAÇÕES E DISCUSSÃO

6.1 Testes e Comparações com o Expert SINTA

Com o objetivo de validar o sistema proposto, dois problemas foram implementados usando o SSEF e o Expert SINTA, de modo a serem feitos testes e comparações. 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: o diagnóstico de abdômen agudo em recém-nascido. A base de conhecimento deste último exemplo foi implementada por LIMA (2000).

Para ambos os sistemas, procurou-se implementar as mesmas regras e variáveis, sempre procurando corresponder os valores de verdade utilizados por este *shell* com os fatores de certeza utilizados pelo Expert SINTA. O número de regras variou, devido a limitações de um ou outro *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.

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 SSEF, os valores de verdade variam de 0 a 1.

- **Primeiro problema: O problema dos vinhos**

Este é um exemplo de conhecimento empírico. Consiste em se escolher os melhores vinhos 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 utilizada para a implementação deste exemplo foi tirado do livro de ARARIBÓIA (1989). A implementação feita utilizando-se o SSEF encontra-se no Anexo 2. No Anexo 3, encontra-se a implementação feita utilizando-se o Expert SINTA.

Neste caso, foram realizados todos os testes para todos os casos possíveis, utilizando-se tanto o Expert SINTA quanto o *shell* implementado. A tabela com os resultados obtidos na realização destes testes pode ser vista no Anexo 6.

A Tabela 6.1 mostra todas as variáveis relacionadas ao problema, com os valores que ela pode assumir.

Tabela 6.1 Variáveis do Sistema Especialista Vinhos.

Variável	Valores Possíveis
Prato Principal	Vegetariano, Carne, Ave, Peixe
Tem Vitela	Sim, Não
Tem Peru	Sim, Não
Tem molho	Sim, Não
Molho	Temperado, Tomate, Doce
Docura Preferida	Suave, Seco, Doce
Cor Preferida	Tinto, Branco

O número total de casos para este problema é 144. Para a realização dos testes, considerou-se que as respostas fornecidas pelo usuário foram dadas com 1 de valor de verdade 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.

Em 10 casos, as respostas obtidas foram idênticas do ponto de vista qualitativo, conforme mostra a Tabela 6.2. No restante dos casos, o SSEF ofereceu mais algumas opções como resposta. Por que isto ocorre?

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 SSEF, 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 0,9 de valor de verdade. O SSEF não rejeita a regra, mas passa a considerar, que

corpreferida=branco com 0,1 de valor de verdade, ou seja, 1 - 0,9.

Tabela 6.2 SE Vinhos - Respostas obtidas idênticas para os dois *shell*.

Variáveis	SSEF		Expert SINTA	
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Branco	Chablis Chardonay Sauvignon Blanc Soave	0,820 0,753 0,742 0,630	Chablis Chardonay Sauvignon Blanc Soave	47,837 40,788 40,284 31,724
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Tinto	Chablis Chardonay Sauvignon Blanc Soave	0,653 0,570 0,548 0,434	Chablis Chardonay Sauvignon Blanc Soave	34,048 29,03 28,672 22,579

Esta consideração foi feita, pois se não há certeza absoluta de uma verdade, há alguma certeza de sua negação. Considerando-se que certeza(tinto) + certeza(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. A Tabela 6.3 mostra alguns resultados onde o SSEF forneceu mais algumas respostas além das obtidas pelo Expert SINTA.

Tabela 6.3 SE Vinhos – Obtenção de outras respostas além das obtidas pelo Expert SINTA.

Variáveis	SSEF		Expert SINTA	
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Branco	Chablis Sauvignon Blanc Chardonay Soave	0,975 0,941 0,138 0,108	Chablis Sauvignon Blanc	75,303 63,413
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,871 0,794 0,138 0,108	Chablis Sauvignon Blanc	57,264 48,223
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Branco	Chablis Sauvignon Blanc Chardonay Soave	0,820 0,742 0,138 0,108	Chablis Sauvignon Blanc	50,37 42,417
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,653 0,570 0,138 0,108	Chablis Sauvignon Blanc	38,304 32,256

Quando todas as respostas são fornecidas com 1 e 100 de valor de verdade as

respostas dadas são as mesmas. Como exemplo, podemos observar o caso mostrado na Tabela 6.4, onde se considerou uma refeição que tem peixe com molho temperado como prato principal e como preferência do usuário, tem-se o vinho suave e branco.

Tabela 6.4 SE Vinhos - Resultados obtidos usando-se valor de verdade igual a 1.

Expert SINTA		SSEF	
Chablis	41,04	Chablis	0,704
Sauvignon Blanc	34,56	Sauvignon Blanc	0,620

Como podemos observar, as respostas obtidas são iguais, considerando-se do ponto de vista qualitativo. Neste caso, as regras que foram utilizadas para computação da verdade são as mesmas, visto que não ocorre mais o caso mostrado no item anterior, pois a escolha foi feita com 1 de valor de verdade. 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.

Em 19 casos, a ordem das respostas fornecidas pelos dois *shell* não foram idênticas. Destes 19 casos, a primeira resposta fornecida coincidiu em 6 casos, sendo que nos outros 13 casos a resposta com maior valor de verdade foi diferente. A Tabela 6.5 mostra exemplos de onde as respostas foram dadas em ordem diferente, mas com a primeira resposta igual, e a Tabela 6.6 mostra casos onde as respostas foram obtidas em ordem diferente, inclusive para a primeira resposta.

Tabela 6.5 SE Vinhos - Respostas obtidas em ordem diferente, coincidindo a opção mais provável.

Variáveis	SSEF		Expert SINTA	
Prato principal = Carne; Tem vitela = Sim Tem molho = Sim; Molho = Tomate Doçura preferida = Seco Cor preferida = Branco	Riesling Chardonay chenin blanc gamay soave pinot noir zinfandel chablis cabernet sauvignon sauvignon blanc	0,662 0,634 0,612 0,545 0,528 0,507 0,485 0,190 0,171 0,161	Riesling Chardonay chenin blanc soave gamay pinot noir zinfandel	29,134 26,261 25,897 20,394 18,662 16,796 15,863
Prato principal = Ave; Tem peru = Não Tem molho = Não Doçura preferida = Doce Cor preferida = Tinto	chablis sauvignon blanc chardonay soave	0,653 0,570 0,548 0,434	chablis chardonay sauvignon blanc soave	38,304 32,659 32,256 25,402

Tabela 6.6 SE Vinhos - Respostas obtidas em ordem diferente, sem coincidir a opção mais provável.

Variáveis	SSEF		Expert SINTA	
Prato principal = Ave Tem peru = Sim Tem molho = Sim Molho = Temperado Doçura preferida = Doce Cor preferida = Branco	pinot noir zinfandel cabernet sauvignon gamay	0,612 0,588 0,566 0,412	cabernet sauvignon pinot noir zinfandel	23,501 22,395 21,151
Prato principal = Ave Tem peru = Sim Tem molho = Sim Molho = Temperado Doçura preferida = Doce Cor preferida = Tinto	pinot noir zinfandel cabernet sauvignon gamay	0,740 0,715 0,700 0,412	cabernet sauvignon pinot noir zinfandel	33,019 31,465 29,717

De forma a quantificar o desempenho do sistema SSEF, torna-se necessário, primeiramente, ter um *padrão ouro*. Segundo, definir um parâmetro que se utilize de uma porcentagem de acertos sobre um conjunto de teste. Neste sentido, far-se-á a assunção de que o especialista implementado utilizando-se do shell Expert SINTA seja o *padrão ouro*. Ou seja, ele sempre fornecerá a melhor resposta a um questionamento. E, o parâmetro de avaliação, que será aqui chamado de *concordância*, será dado por:

$$concordância = \left(\frac{número\ de\ acertos}{número\ total\ de\ casos} \right) * 100$$

De acordo com o que foi apresentado, observa-se que existe uma concordância em 86,8 % de todas as possibilidades de questionamentos e respostas. Em 6,94 % das consultas, o sistema SSEF responde exatamente da mesma forma que o Expert SINTA. Em 79,86 % dos casos, o sistema SSEF, além de concordar com todas as respostas do Expert SINTA fornece, ainda, outras alternativas de respostas.

Consequentemente, em 13,19 % de possibilidades não existe concordância em todas as respostas dadas pelo Expert Sinta. No entanto, em 4,17 % de todas as possíveis perguntas e respostas, há concordância na resposta mais provável.

Do ponto de vista qualitativo, todas as respostas apresentadas pelo SSEF, mesmo aquelas que não apresentam concordância com o Expert SINTA, foram consideradas boas soluções, dado que a solução apresentada pelo Expert SINTA foi considerada como "melhor solução".

- **Segundo problema: Diagnóstico de abdômen agudo em recém nascido**

O SE para diagnóstico de abdômen agudo foi desenvolvido em uma disciplina ministrada na Pós-Graduação em Medicina, pelo mestrando Angevaldo LIMA (2000), que é cirurgião-pediátrico.

A motivação para o desenvolvimento do sistema foi o fato do diagnóstico preciso do abdômen agudo durante o período neonatal, quando possível, ser tarefa das mais difíceis, dentro da cirurgia pediátrica. A dificuldade de se obter informações precisas, quer por causa das peculiaridades da faixa etária quer pela simples ausência de queixas ativas, já que a criança não reclama, torna o exercício diagnóstico um exemplo de raciocínio clínico nem sempre coroado de êxito.

O modelo se destina a fornecer uma avaliação aproximada das possibilidades diagnósticas, respeitadas as peculiaridades de cada síndrome. Por motivos de limitação de objetivos, o SE implementado restringiu-se àquelas doenças que afetam os órgãos do tubo digestivo, sendo excluídas as doenças do aparelho urinário e não se aprofundando na origem dos processos neoplásicos. É importante também frisar que o programa avalia apenas problemas relacionados às doenças intra-abdominais, não sendo possível correlacionar sinais ou sintomas originados por outras doenças concomitantes ou secundários à patologia clínica que se manifeste como "pseudo abdômen agudo".

O trabalho faz o diagnóstico entre as seguintes possibilidades: Síndrome Inflamatória, Síndrome Obstrutiva, Síndrome Perfurativa ou Síndrome Hemorrágica.

O SE foi inicialmente implementado utilizando-se do *shell* Expert SINTA. Após ter sido validado pelo especialista que o criou e pelos outros médicos que também cursavam a disciplina, este foi implementado usando-se o SSEF. Para a implementação feita com o Expert SINTA, considerou-se que não deveria ser atribuído fator de certeza para as respostas dadas pelo usuário. Ou seja, o fator de certeza é considerado sempre 100. Na implementação feita usando-se o SSEF, de modo a fazer a correspondência com o Expert SINTA, as respostas foram consideradas *crisp*, ou seja, não é atribuído um valor de verdade às respostas dadas pelo usuário, sendo estas sempre consideradas com valor 1. A base de conhecimento utilizada pelo SSEF encontra-se no Anexo 4. No Anexo 5, encontra-se a implementação feita usando o Expert SINTA. Para este exemplo, serão mostrados apenas dois casos de situações reais para ilustração.

Caso 1: O primeiro caso implementado foi um caso de diagnóstico mais provável de síndrome obstrutiva, caracterizado principalmente pela ausência da eliminação do mecônio e pela presença de mal formação anorretal. Estas duas características apontam somente para o caso de síndrome obstrutiva. As outras características apontadas foram: vômitos precoces (que é característico da síndrome obstrutiva e também das síndromes inflamatória e perfurativa), distensão de abdômen superior (que é característico das síndromes obstrutiva e inflamatória), ausência de desconforto respiratório e ausência de icterícia (sendo que estes dois últimos não são característicos de nenhuma das síndromes apresentadas). Como já mencionado anteriormente, as variáveis são do tipo *crisp*.

Os resultados obtidos com a implementação de um e outro *shell* são mostrados na Tabela 6.7.

Tabela 6.7 Resultados obtidos: Diagnóstico de Abdômen Agudo – Exemplo 1.

Expert SINTA		SSEF	
Obstrutiva	99,925	Obstrutiva	0,999
Inflamatória	52	Inflamatória	0,52
Perfurativa	30	Perfurativa	0,3

Como se pode observar, os dois *shell* apresentam respostas idênticas para o quadro apresentado.

Caso 2: O segundo caso implementado informa sintomas que são característicos das síndromes inflamatória e perfurativa. Os sintomas apresentados ao SE foram: vômitos ocasionais (característico das 4 síndromes), distensão leve do abdômen (característico das 4 síndromes), toque retal acompanhado de dor (característico das síndromes perfurativa e inflamatória, sendo que mais da perfurativa), eliminação normal de mecônio, ausência de desconforto respiratório e ausência de icterícia (sendo que estes três últimos não são característicos de nenhuma das síndromes apresentadas).

Os resultados obtidos nesta consulta estão apresentados na Tabela 6.8.

Tabela 6.8 Resultados obtidos: Diagnóstico de Abdômen Agudo – Exemplo 2.

Expert SINTA		SSEF	
Perfurativa	92,5	Perfurativa	0,925
Inflamatória	90	Inflamatória	0,9
Hemorrágica	75	Hemorrágica	0,75
Obstrutiva	44	Obstrutiva	0,44 ou 44

Também nestes caso, a resposta obtida foi idêntica para os dois *shell*. E em vários outros casos implementados, as respostas foram sempre idênticas. Infelizmente, por falta de um especialista médico consultor não foi possível se fazer uma análise comparativa como aquela feita no problema anterior. Melhor ainda, neste caso – um caso na área médica, o ideal seria um estudo de sensibilidade, especificidade e eficiência (as definições para estes conceitos podem ser encontradas em LOPES (1996)). Por conseguinte, apesar de impossível uma validação formal do sistema, qualitativamente, ambos os sistemas apresentaram o mesmo comportamento.

Comparando os resultados obtidos com os dois problemas implementados, observa-se que as respostas foram idênticas no segundo problema. No primeiro problema, onde foi utilizado valor de verdade igual a 1, as respostas não foram numericamente iguais. Isso ocorre pelo fato deste primeiro problema apresentar variáveis intermediárias, ou seja, que são inferidas a partir de outras regras, enquanto o segundo problema não possui este tipo de variável. Os valores obtidos pelo Expert SINTA para estas variáveis intermediárias são diferentes dos valores obtidos pelo SSEF, fazendo com que o valor obtido para a conclusão não seja o mesmo.

No entanto, deve-se deixar bem claro, que o objetivo ao se realizar estas experimentações, era comparar qualitativamente os valores obtidos por um e outro *shell*, de forma a validar os resultados fornecidos pelo *shell* apresentado. Como são duas abordagens completamente diferentes, já que uma usa fatores de certeza e a outra utiliza lógica fuzzy, não se esperava que fornecessem respostas idênticas do ponto de vista quantitativo.

Concluindo-se, pode ser observado que os resultados, ao menos do ponto de vista qualitativo, coincidem, sendo os resultados obtidos pelo SSEF considerados satisfatórios. No entanto, o Expert SINTA apresenta algumas características que ainda não são possíveis de ser implementadas pelo SSEF. Um exemplo disto, é o fato do Expert SINTA permitir várias metas, ou seja, após chegar a uma primeira meta o sistema continua em modo consulta até que todas a metas tenham sido atingidas. Isto

também é possível de implementar no *shell* proposto, desde que realizadas algumas modificações de interface

O Expert SINTA apresenta vantagens sobre o SSEF referentes a facilidades oferecidas pela interface. Neste sentido, deve ainda ser realizado muito trabalho de modo que os dois apresentem as mesmas facilidades para montagem de um SE.

Por outro lado, 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. No caso do SSEF, esta situação não ocorre.

Outra diferença do *shell* 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. É uma outra opção para um desenvolvedor que queira construir um SE, sendo que ele irá precisar do conhecimento a respeito do problema e a partir disso montar o SE utilizando-se das duas formas e verificando qual fornece resultados mais satisfatórios.

Pelo fato da máquina de inferência ter sido implementada como uma dll, isto permite que este arquivo seja utilizado em conjunto com qualquer outra aplicação. Ou seja, desde que a base de conhecimento seja criada a partir do *shell* implementado, qualquer outro sistema pode realizar uma consulta a essa base, sendo necessário apenas ligar a dll criada ao sistema em questão. Para isto, se utilizaria a mesma função pública que o próprio *shell* usa para realizar a consulta, ou seja, a função *consultar*, que tem como parâmetros o nome do arquivo que se está consultando e tem como valor de retorno o valor zero ou um, indicando se a dll foi executada com sucesso ou se algum erro ocorreu durante a sua execução.

6.2 Trabalhos Futuros

De forma a se obter um sistema "utilizável" algum trabalho ainda se faz necessário. Por exemplo:

- Utilizar outras equações para obter o valor de verdade final para uma determinada conclusão;
- Inserir no sistema mais facilidades em relação à interface, verificando-se, entre os pesquisadores do GPEB, quais são as principais necessidades existentes;
- Permitir que o SSEF aceite mais de uma meta, como no Expert SINTA.

No entanto, não se acredita que tais desenvolvimentos se constituam em trabalho de pesquisa, mas sim em trabalho à nível de iniciação científica, posto que a parte de pesquisa e desenvolvimento, considerando-se o sistema proposto, já está resolvida.

ANEXO1 – LISTAGEM DA MÁQUINA DE INFERÊNCIA

```
/*************************************************************
                Copyright (c) UFSC
 Project:  MAQUINAINFERENCIA
 FileName: MAQUINAINFERENCIA.PRO
 Purpose: No description
 Written by: Maria Nazaré Munari Angeloni
 Comments:
*************************************************************************/

include "maquinainferencia.inc"
include "maquinainferencia.con"
include "hlptopic.con"

GLOBAL DOMAINS

LETRA = CHAR
INTEIRO = INTEGER
PALAVRA = STRING
LISTA = PALAVRA*
LISTAREAL = REAL*
LISTINT = INTEIRO*

PERGCRISP = procedure (INTEIRO,STRING,REAL) - (i,i,o) language stdcall /*Declaration of the Dll
Delphi procedure*/
PERGFUZZY = procedure (INTEIRO,STRING,REAL) - (i,i,o) language stdcall /*Declaration of the Dll
Delphi procedure*/

GLOBAL DATABASE

explic_modif(INTEIRO,LISTA)
regra_modificada(INTEIRO,PALAVRA,PALAVRA,PALAVRA,REAL,LISTA)
perg_modif(INTEIRO,PALAVRA,LISTA,PALAVRA,LISTA,LISTA,PALAVRA)
fator_certeza(PALAVRA,PALAVRA,LISTA,LISTA)
escolha(INTEIRO,INTEIRO)
introducao(LISTA)
ajuda(INTEIRO,LISTA)
especialista(LISTA)
nome_do_arquivo(PALAVRA,PALAVRA)
meta(PALAVRA)
certeza_somada(INTEIRO,PALAVRA,REAL)
dadoreal(REAL)
dadoint(INTEIRO)
dadoext(INTEIRO)
maior_certeza(PALAVRA,REAL)
pergunta_numero(INTEIRO)
nomedoarquivo(PALAVRA)
num_pre(INTEIRO,INTEIRO)
historia(LISTA)
juntos(PALAVRA,PALAVRA)
pergunta(INTEIRO,PALAVRA,LISTA,PALAVRA,LISTA,LISTA,PALAVRA)
regra(INTEIRO,PALAVRA,PALAVRA,PALAVRA,REAL,LISTA)
```

```
explicacao(INTEIRO,LISTA)
parametro_janela (INTEIRO,INTEIRO,INTEIRO, INTEIRO,INTEIRO,INTEIRO,INTEIRO)
certeza_fixada(PALAVRA,REAL)
resposta(PALAVRA,PALAVRA,REAL)

GLOBAL PREDICATES

procedure consultar (string NomeDoArquivo, integer ValorRetorno) - (i,o) language stdcall
/*Procedimento que será exportado para o programa Delphi*/

nondeterm emite_opiniao(PALAVRA,PALAVRA,PALAVRA,REAL) - (i,i,i,i) (o,i,o,o) (i,i,i,o)
acerta_certeza(PALAVRA,REAL,REAL)-(i,i,o)
completa(LISTA,LISTA) - (i,o)
acrescenta(INTEIRO,LISTA,LISTA) - (i,i,o)
nondeterm determina(PALAVRA,PALAVRA,PALAVRA,REAL,REAL) - (o,o,i,o,i) (i,i,i,i,i) (i,o,i,o,i)
(i,i,i,o,i)
nondeterm acha_resposta(PALAVRA)
mostra_resposta
retorna_ligado(PALAVRA,PALAVRA,PALAVRA) - (i,i,o)
resposta_ligada(PALAVRA,PALAVRA,PALAVRA,REAL,PALAVRA,REAL) - (i,i,i,o,i,i)
nondeterm chama_especialista(STRING, INTEIRO)
junta_certezas(PALAVRA)
resolve(PALAVRA)
acerta_maior(PALAVRA,REAL,PALAVRA,REAL,PALAVRA,REAL)
ordena
nondeterm lacoexterno
lacointerno(INTEIRO)
troca(INTEIRO,INTEIRO)
faz_pergunta(LISTA)
pergunta_logo(INTEIRO,PALAVRA,PALAVRA,REAL) - (i,i,i,i) (i,o,o,o) (i,i,o,o) (i,i,i,o)
index(LISTA,INTEIRO,PALAVRA)
falha

CLAUSES

consultar (NomeDoArquivo, ValorRetorno):-
        Stackmark = mem_MarkGStack(),
        dlg_Note (NomeDoArquivo),
        ValorRetorno = 1,
        chama_especialista(NomeDoArquivo, ValorRetorno) ,!.

chama_especialista("", ValorRetorno) :- ValorRetorno = 0.
chama_especialista(NomeDoArquivo, ValorRetorno) :-
        consult(NomeDoArquivo), %Aqui deve consultar o nome do arquivo que vem do Delphi
        save ("MeuArquivo.txt"),
        meta(ASSUNTO), !,
        dlg_Note(ASSUNTO), !,
        assert(pergunta_numero(1)),
        historia(L),
        faz_pergunta(L),
        acha_resposta(ASSUNTO),
        ValorRetorno = 1,!.

acha_resposta(ASSUNTO) :-
    emite_opiniao(S,ASSUNTO,RESPOSTA,CERTEZA),
        acerta_certeza(S,CERTEZA,CERTCORR),
        assert(certeza_fixada(RESPOSTA,CERTCORR)),
        falha,!.
```

```
acha_resposta(ASSUNTO) :-
        elimina_dadoint,
        assert(dadoint(0)),
        junta_certezas(ASSUNTO),
/*  So muda esta linha para ordenar */
        ordena,
        save ("resposta.txt").
        %Aqui serão mostradas as respostas do sistema. O controle volta para o Delphi

acha_resposta(_).

/*  predicados para ordenacao    */

ordena :-
        dadoint(N),
        elimina_dadoint,
        assert(dadoext(N)),
        lacoexterno,!.

lacoexterno :-
        dadoext(N), N > 1, assert(dadoint(1)),
        lacointerno(N), retract(dadoext(_)),
        N1 = N - 1, assert(dadoext(N1)),
        lacoexterno,!.
lacoexterno :- retract(dadoext(_)).
lacoexterno.

lacointerno(N) :-
        dadoint(J), K = J + 1, J < N, troca(J,K),
        retract(dadoint(_)), assert(dadoint(K)),
        lacointerno(N),!.
lacointerno(_) :- elimina_dadoint.

troca(J,K) :-
        certeza_somada(J,AJ,BJ), certeza_somada(K,AK,BK), BJ < BK,
        retract(certeza_somada(J,AJ,BJ)),
        retract(certeza_somada(K,AK,BK)),
        assert (certeza_somada(K,AJ,BJ)),
        assert (certeza_somada(J,AK,BK)),!.
troca(J,K) :-
        certeza_somada(J,_,BJ), certeza_somada(K,_,BK), BJ >= BK,  !.


/*  fim da ordenacao          */

junta_certezas(ASSUNTO) :-
        certeza_fixada(Resposta,_),
        elimina_dadoreal,
        assert(dadoreal(0.0)),
        resolve(Resposta),
        junta_certezas(ASSUNTO),!.
junta_certezas(ASSUNTO) :- bound(ASSUNTO),!.

resolve(R) :-
        certeza_fixada(R,C),
        retract(certeza_fixada(R,C)),
        dadoreal(C1), C2 = C + C1 - C * C1,
        retract( dadoreal(_)), assert(dadoreal(C2)),
        resolve(R),!.
```

```
resolve(R) :-
        dadoreal(C),
        dadoint(N2),
        elimina_dadoint,
        N1 = N2 + 1,
        assert(dadoint(N1)),
        retract(dadoreal(_)),
        assert(certeza_somada(N1,R,C)),!.

acerta_maior(R,C,R1,C1,R,C)  :- bound(R1), C >= C1,!.
acerta_maior(R,C,R1,C1,R1,C1) :- bound(R),  C <  C1,!.

emite_opiniao("","","",1.0) :- !.
emite_opiniao(SIMOUNAO,ASSUNTO,RESPOSTA,CERTEZA) :-
        bound(RESPOSTA),
        regra(_,SIMOUNAO,ASSUNTO,"X",Cert_Regra,LISTA_SE),
        completa(LISTA_SE,LISTACHEIA),
        LISTACHEIA =
[SN1,P1,R1,SN2,P2,R2,SN3,P3,R3,SN4,P4,R4,SN5,P5,R5,SN6,P6,R6,SN7,P7,R7,SN8,P8,R8,SN9,P9,R
9,SN10,P10,R10],
        resposta_ligada(SN1,P1,R1,C1,RESPOSTA,1.0),
        resposta_ligada(SN2,P2,R2,C2,RESPOSTA,C1),
        resposta_ligada(SN3,P3,R3,C3,RESPOSTA,C2),
        resposta_ligada(SN4,P4,R4,C4,RESPOSTA,C3),
        resposta_ligada(SN5,P5,R5,C5,RESPOSTA,C4),
        resposta_ligada(SN6,P6,R6,C6,RESPOSTA,C5),
        resposta_ligada(SN7,P7,R7,C7,RESPOSTA,C6),
        resposta_ligada(SN8,P8,R8,C8,RESPOSTA,C7),
        resposta_ligada(SN9,P9,R9,C9,RESPOSTA,C8),
        resposta_ligada(SN10,P10,R10,C10,RESPOSTA,C9),
        acerta_certeza(SIMOUNAO,C10,C11),
        CERTEZA = Cert_Regra * C11 .

emite_opiniao(SimNao,ASSUNTO,RESPOSTA,CERTEZA) :-
        bound(RESPOSTA),
        pergunta(_,_,_,_,_,ASSUNTO),
        resposta(ASSUNTO,RESPOSTA1,CERTEZA1),!,
        determina(SimNao,RESPOSTA,RESPOSTA1,CERTEZA,CERTEZA1),
        CERTEZA > 0 ,!.

emite_opiniao(S,A,R,C) :-
        pergunta(N,_,_,_,_,A),
        pergunta_logo(N,S,R,C),
        C > 0 ,!.

emite_opiniao(SIMOUNAO,ASSUNTO,RESPOSTA,CERTEZA) :-
        bound(RESPOSTA),
        regra(_,SIMOUNAO,ASSUNTO,RESPOSTA,Cert_Regra,LISTA_SE),
        completa(LISTA_SE,LISTACHEIA),
        LISTACHEIA =
[SN1,P1,R1,SN2,P2,R2,SN3,P3,R3,SN4,P4,R4,SN5,P5,R5,SN6,P6,R6,SN7,P7,R7,SN8,P8,R8,SN9,P9,R
9,SN10,P10,R10],
        emite_opiniao(SN1,P1,R1,C1),
        emite_opiniao(SN2,P2,R2,C2), minimo(C1,C2,C21),
        emite_opiniao(SN3,P3,R3,C3), minimo(C21,C3,C31),
        emite_opiniao(SN4,P4,R4,C4), minimo(C31,C4,C41),
        emite_opiniao(SN5,P5,R5,C5), minimo(C41,C5,C51),
        emite_opiniao(SN6,P6,R6,C6), minimo(C51,C6,C61),
        emite_opiniao(SN7,P7,R7,C7), minimo(C61,C7,C71),
```

```
        emite_opiniao(SN8,P8,R8,C8), minimo(C71,C8,C81),
        emite_opiniao(SN9,P9,R9,C9), minimo(C81,C9,C91),
        emite_opiniao(SN10,P10,R10,C10), minimo(C91,C10,Ct1),
        acerta_certeza(SIMOUNAO,Ct1,Ct2),
        CERTEZA = Cert_Regra * Ct2 .

emite_opiniao(SIMOUNAO,ASSUNTO,RESPOSTA,CERTEZA) :-
        meta(ASSUNTO),
        regra(_,SIMOUNAO,ASSUNTO,RESPOSTA,Cert_Regra,LISTA_SE),
        completa(LISTA_SE,LISTACHEIA),
        LISTACHEIA                                                    =
[SN1,P1,R1,SN2,P2,R2,SN3,P3,R3,SN4,P4,R4,SN5,P5,R5,SN6,P6,R6,SN7,P7,R7,SN8,P8,R8,SN9,P9,R
9,SN10,P10,R10],
        emite_opiniao(SN1,P1,R1,C1),
        emite_opiniao(SN2,P2,R2,C2), minimo(C1,C2,C21),
        emite_opiniao(SN3,P3,R3,C3), minimo(C21,C3,C31),
        emite_opiniao(SN4,P4,R4,C4), minimo(C31,C4,C41),
        emite_opiniao(SN5,P5,R5,C5), minimo(C41,C5,C51),
        emite_opiniao(SN6,P6,R6,C6), minimo(C51,C6,C61),
        emite_opiniao(SN7,P7,R7,C7), minimo(C61,C7,C71),
        emite_opiniao(SN8,P8,R8,C8), minimo(C71,C8,C81),
        emite_opiniao(SN9,P9,R9,C9), minimo(C81,C9,C91),
        emite_opiniao(SN10,P10,R10,C10), minimo(C91,C10,Ct1),
        acerta_certeza(SIMOUNAO,Ct1,Ct2),
        CERTEZA = Cert_Regra * Ct2 .

mostra_resposta.

acerta_certeza("Sim",Certeza1,Certeza1) :- !.

acerta_certeza("Nao",Certeza1,Certeza) :- !,Certeza = 1 - Certeza1 , !.

completa(LISTA_SE,LISTACHEIA) :- comprlista(LISTA_SE,Compr),
        Falta = 30 - Compr ,
        acrescenta(Falta,LISTA_SE,LISTACHEIA).

acrescenta(0,LISTA_SE,LISTA_SE) :-!.
acrescenta(Falta,LISTA_SE,LISTACHEIA) :- !,
        Falta > 0,
        Falta1 = Falta - 3,
        junta_lista(LISTA_SE,["","",""],LISTAINT),
        acrescenta(Falta1,LISTAINT,LISTACHEIA).

determina("Sim",RESPOSTA,RESPOSTA,CERTEZA,CERTEZA).
determina("Sim",RESPOSTA,RESPOSTA1,CERTEZA,CERTEZA1) :-
        bound(RESPOSTA), not(RESPOSTA = RESPOSTA1),
        CERTEZA = 1 - CERTEZA1.
determina("Nao",RESPOSTA,RESPOSTA,CERTEZA,CERTEZA1) :-
        CERTEZA = 1 - CERTEZA1.
determina("Nao",RESPOSTA,RESPOSTA1,CERTEZA,CERTEZA) :-
        bound(RESPOSTA), not(RESPOSTA = RESPOSTA1).

retorna_ligado("X",RESPOSTA,RESPOSTA) :- ! , bound(RESPOSTA).
retorna_ligado(R1,RESPOSTA,R1) :-
        bound(R1), bound(RESPOSTA), not (R1 ="X"),!.
retorna_ligado(_,_,"") :-
        write("ligado errado").

resposta_ligada("","","",C,_,C) :- !.
```

```
resposta_ligada(SN,P,R,C,RESP,Ca) :-
        retorna_ligado(R,RESP,L),
        emite_opiniao(SN,P,L,Cn), minimo(Ca,Cn,C),!.

faz_pergunta([]).
faz_pergunta([C|L]) :-
        str_int(C,N),
        pergunta_logo(N,"Sim",_,_),
        faz_pergunta(L).

pergunta_logo(N,SN,R,C) :-
        pergunta(N,T,L,"",[],LP,A),
        not(resposta(A,_,_)),
        explicacao(N,E),
                Stackmark = mem_MarkGStack(),
                trap(DllHandle = vpi_LoadDll("ObterRespostas.dll"),_,fail),/*My Delphi DLL -
GetValueDll*/
                  DllHandle>0,
                mem_ReleaseGStack(Stackmark),
            trap (PerguntaCrisp = cast(PERGCRISP, vpi_GetDllProc( DllHandle, "perguntaCrisp"
)),_,fail),
                mem_ReleaseGStack(Stackmark),
                mem_ReleaseGStack(Stackmark),
                str_len(R1,50),
                 trap(PerguntaCrisp(N,R1,C2),_,dlg_note("First trap")),/*Get the first value*/
                 vpi_FreeDll(DllHandle),/* Return to Exe Delphi file*/
                assert(resposta(A,R1,C2)),
                determina(SN,R,R1,C,C2),
                !.

pergunta_logo(N,SN,R,C) :-
        pergunta(N,T,L,NC,LC,LP,A),
        not(resposta(A,_,_)),
        explicacao(N,E),
        Stackmark = mem_MarkGStack(),
        trap(DllHandle = vpi_LoadDll("ObterRespostas.dll"),_,fail),/*My Delphi DLL - GetValueDll*/
        DllHandle>0,
        mem_ReleaseGStack(Stackmark),
        trap (PerguntaFuzzy = cast(PERGFUZZY, vpi_GetDllProc( DllHandle, "perguntaFuzzy"
)),_,fail),
        mem_ReleaseGStack(Stackmark),
        mem_ReleaseGStack(Stackmark),
        str_len (R1,100), %ak
        trap(PerguntaFuzzy(N,R1,C1),_,dlg_note("First trap")),/*Get the first value*/
         vpi_FreeDll(DllHandle),/* Return to Exe Delphi file*/
        assert(resposta(A,R1,C1)),
        determina(SN,R,R1,C,C1),
        !.

GOAL

 true.
```

ANEXO 2 – BASE DE CONHECIMENTO I – VINHOS

```
introducao(["Eu sou um especialista em Vinhos.","Espero que eu possa ajuda'-lo em sua","escolha de um vinho apropriado para o","seu jantar."])
meta("VINHO")
fator_certeza("fator1","Tem certeza",["Sim","Nao","Nao Sei"],["1","0","0.5"])
fator_certeza("fator2","Tem certeza",["Certeza Absoluta","Quase Certo","Mais ou menos","Nao sei"],["0.9","0.7","0.6","0.5"])
pergunta(1,"Qual o prato principal?",["Vegetariano","Ave","Peixe","Carne"],"Tem certeza",["Sim","Nao","Nao Sei"],["1","0","0.5"],"prato principal")
pergunta(2,"Este prato tem vitela?",["Sim","Nao"],"Tem certeza",["Sim","Nao","Nao Sei"],["1","0","0.5"],"tem vitela")
pergunta(3,"Este prato tem peru?",["Sim","Nao"],"Tem certeza",["Sim","Nao","Nao Sei"],["1","0","0.5"],"tem peru")
pergunta(4,"Este prato tem molho?",["Sim","Nao"],"Tem certeza",["Sim","Nao","Nao Sei"],["1","0","0.5"],"tem molho")
pergunta(5,"Como e o molho?",["Tomate","Doce","Temperado"],"Tem certeza",["Certeza Absoluta","Quase Certo","Mais ou menos","Nao sei"],["0.9","0.7","0.6","0.5"],"molho")
pergunta(6,"Como prefere o vinho?",["Seco","Suave","Doce"],"Tem certeza",["Certeza Absoluta","Quase Certo","Mais ou menos","Nao sei"],["0.9","0.7","0.6","0.5"],"docura preferida")
pergunta(7,"De que cor prefere o vinho?",["Branco","Tinto"],"Tem certeza",["Certeza Absoluta","Quase Certo","Mais ou menos","Nao sei"],["0.9","0.7","0.6","0.5"],"cor preferida")
regra(1,"Sim","VINHO","gamay",0.9,["Sim","cor recomendada","Tinto","Sim","docura recomendada","Doce"])
regra(2,"Sim","VINHO","chablis",0.95,["Sim","cor recomendada","Branco","Sim","docura recomendada","Seco"])
regra(3,"Sim","VINHO","cabernet sauvignon",0.85,["Sim","cor recomendada","Tinto","Sim","docura recomendada","Seco"])
regra(4,"Sim","VINHO","riesling",0.9,["Sim","cor recomendada","Branco","Sim","docura recomendada","Doce"])
regra(5,"Sim","VINHO","sauvignon blanc",0.8,["Sim","cor recomendada","Branco","Sim","docura recomendada","Seco"])
regra(6,"Sim","VINHO","chenin blanc",0.8,["Sim","cor recomendada","Branco","Sim","docura recomendada","Doce"])
regra(7,"Sim","VINHO","pinot noir",0.9,["Sim","cor recomendada","Tinto","Sim","docura recomendada","Suave"])
regra(8,"Sim","VINHO","soave",0.7,["Sim","cor recomendada","Branco","Sim","docura recomendada","Suave"])
regra(9,"Sim","VINHO","chardonay",0.9,["Sim","cor recomendada","Branco","Sim","docura recomendada","Suave"])
regra(10,"Sim","VINHO","zinfandel",0.85,["Sim","cor recomendada","Tinto","Sim","docura recomendada","Suave"])
regra(11,"Sim","cor recomendada","X",1,["Sim","cor preferida","X","Sim","melhor cor","X"])
regra(12,"Sim","cor recomendada","X",1,["Sim","prato principal","Vegetariano","Sim","cor preferida","X"])
regra(13,"Sim","cor recomendada","Tinto",0.8,["Sim","melhor cor","Tinto"])
regra(14,"Sim","cor recomendada","Branco",0.8,["Sim","melhor cor","Branco"])
regra(15,"Sim","melhor cor","Tinto",0.9,["Sim","prato principal","Carne","Sim","tem vitela","Nao"])
regra(16,"Sim","melhor cor","Branco",0.8,["Sim","prato principal","Carne","Sim","tem vitela","Sim"])
regra(17,"Sim","melhor cor","Branco",0.9,["Sim","prato principal","Peixe"])
regra(18,"Sim","melhor cor","Tinto",0.8,["Sim","prato principal","Ave","Sim","tem peru","Sim"])
regra(19,"Sim","melhor cor","Branco",0.9,["Sim","prato principal","Ave","Nao","tem peru","Sim"])
regra(20,"Sim","melhor cor","Tinto",0.8,["Nao","prato principal","Peixe","Sim","tem molho","Sim","Sim","molho","Tomate"])
```

```
regra(21,"Sim","docura recomendada","X",1,["Sim","docura preferida","X","Sim","melhor docura","X"])
regra(22,"Sim","docura recomendada","X",0.7,["Sim","prato principal","Vegetariano","Sim","docura
preferida","X"])
regra(23,"Sim","docura recomendada","Doce",0.8,["Sim","melhor docura","Doce"])
regra(24,"Sim","docura recomendada","Seco",0.8,["Sim","melhor docura","Seco"])
regra(25,"Sim","docura recomendada","Suave",0.8,["Sim","melhor docura","Seco","Sim","docura
preferida","Doce"])
regra(26,"Sim","docura recomendada","Suave",0.8,["Sim","melhor docura","Doce","Sim","docura
preferida","Seco"])
regra(27,"Sim","docura recomendada","Suave",0.8,["Sim","melhor docura","Suave"])
regra(28,"Sim","melhor docura","Doce",0.9,["Sim","tem molho","Sim","Sim","molho","Doce"])
regra(29,"Sim","melhor docura","Seco",0.6,["Sim","tem molho","Sim","Sim","molho","Temperado"])
regra(30,"Sim","melhor docura","Seco",0.7,["Sim","tem molho","Nao"])
regra(31,"Sim","melhor docura","Doce",0.5,["Sim","tem molho","Sim","Sim","molho","Tomate"])
explicacao(1,["Para recomendar um vinho, preciso","recomendar uma docura e uma
cor","apropriadas. Para isto, preciso saber","qual o prato principal de seu jantar."])
explicacao(2,["Para recomendar uma cor e uma docura,","preciso saber de que consta seu prato","de
carne."])
explicacao(3,["O peru e a unica ave que pede um","vinho tinto, por isso preciso saber","ha peru neste
prato."])
explicacao(4,["O molho e importante na escolha de","um vinho, por isso preciso saber se","tem molho no
prato e qual o tipo de","molho."])
explicacao(5,["O molho e importante na escolha de","um vinho, por isso preciso saber se","tem molho no
prato e qual o tipo de","molho."])
explicacao(6,["7 - Para recomendar uma das marcas","conhecidas de vinho,","preciso recomendar a
cor","e a docura do referido vinho."])
explicacao(7,["6 - A fim de determinar a melhor cor","e a melhor doc para o vinho","que vai acompanhar
o seu jantar,","preciso de todos os dados sobre","o prato principal e sobre","o molho que vai ser usado."])
historia([])
```

ANEXO 3 – BASE DE CONHECIMENTO II – VINHOS

```
---------------------------------------------------------------------------
SOBRE O SISTEMA ESPECIALISTA
 -- Nome:   VINHOS
 -- Autores: MARIA NAZARÉ MUNARI ANGELONI
---------------------------------------------------------------------------

SE corRecomendada = tinto
   E  docuraRecomendada = doce
   ENTÃO Vinho = gamay CNF 90%
SE corRecomendada = branco
   E  docuraRecomendada = seco
   ENTÃO Vinho = chablis CNF 95%
SE corRecomendada = tinto
   E  docuraRecomendada = seco
   ENTÃO Vinho = cabernet Sauvignon CNF 85%
SE corRecomendada = branco
   E  docuraRecomendada = doce
   ENTÃO Vinho = riesling CNF 90%
SE corRecomendada = branco
   E  docuraRecomendada = seco
   ENTÃO Vinho = sauvignon blanc CNF 80%
SE corRecomendada = branco
   E  docuraRecomendada = doce
   ENTÃO Vinho = Chenin blanc CNF 80%
SE corRecomendada = tinto
   E  docuraRecomendada = suave
   ENTÃO Vinho = pinot noir CNF 90%
SE corRecomendada = branco
   E  docuraRecomendada = suave
   ENTÃO Vinho = soave CNF 70%
SE corRecomendada = branco
   E  docuraRecomendada = suave
   ENTÃO Vinho = chardonay CNF 90%
SE corRecomendada = tinto
   E  docuraRecomendada = suave
   ENTÃO Vinho = zinfandel CNF 85%
SE CorPreferida = branco
   E  melhorCor = branco
   ENTÃO corRecomendada = branco CNF 100%
 SE CorPreferida = tinto
   E  melhorCor = tinto
   ENTÃO corRecomendada = tinto CNF 100%
SE pratoPrincipal = Vegetariano
   E  CorPreferida = branco
   ENTÃO corRecomendada = branco CNF 100%
SE pratoPrincipal = Vegetariano
   E  CorPreferida = tinto
   ENTÃO corRecomendada = tinto CNF 100%
SE melhorCor = tinto
   ENTÃO corRecomendada = tinto CNF 80%
SE melhorCor = branco
   ENTÃO corRecomendada = branco CNF 80%
```

SE pratoPrincipal = Carne
  E temVitela = Não
  ENTÃO melhorCor = tinto CNF 90%
SE pratoPrincipal = Carne
  E temVitela = Sim
  ENTÃO melhorCor = branco CNF 80%
SE pratoPrincipal = Peixe
  ENTÃO melhorCor = branco CNF 90%
SE pratoPrincipal = Ave
  E temPeru = Sim
  ENTÃO melhorCor = tinto CNF 80%
SE pratoPrincipal = Ave
  E temPeru = Não
  ENTÃO melhorCor = branco CNF 90%
SE NÃO pratoPrincipal = Peixe
  E temMolho = Sim
  E qualMolho = tomate
  ENTÃO melhorCor = tinto CNF 80%
SE docuraPreferida = doce
  E melhorDocura = doce
  ENTÃO docuraRecomendada = doce CNF 100%
SE docuraPreferida = seco
  E melhorDocura = seco
  ENTÃO docuraRecomendada = seco CNF 100%
SE docuraPreferida = suave
  E melhorDocura = suave
  ENTÃO docuraRecomendada = suave CNF 100%
SE pratoPrincipal = Vegetariano
  E docuraPreferida = suave
  ENTÃO docuraRecomendada = suave CNF 70%
SE pratoPrincipal = Vegetariano
  E docuraPreferida = doce
  ENTÃO docuraRecomendada = doce CNF 70%
SE pratoPrincipal = Vegetariano
  E docuraPreferida = seco
  ENTÃO docuraRecomendada = seco CNF 70%
SE melhorDocura = doce
  ENTÃO docuraRecomendada = doce CNF 80%
SE melhorDocura = seco
  ENTÃO docuraRecomendada = seco CNF 80%
SE melhorDocura = seco
  E docuraPreferida = doce
  ENTÃO docuraRecomendada = suave CNF 80%
SE melhorDocura = doce
  E docuraPreferida = seco
  ENTÃO docuraRecomendada = suave CNF 80%
SE melhorDocura = suave
  ENTÃO docuraRecomendada = suave CNF 80%
SE temMolho = Sim
  E qualMolho = doce
  ENTÃO melhorDocura = doce CNF 90%
SE temMolho = Sim E qualMolho = temperado
  ENTÃO melhorDocura = seco CNF 60%
SE temMolho = Não
  ENTÃO melhorDocura = seco CNF 70%
SE temMolho = Sim E qualMolho = tomate
  ENTÃO melhorDocura = doce CNF 50%

ANEXO 4 – BASE DE CONHECIMENTO I – DIAGNÓSTICO DE ABDÔMEN AGUDO

```
introducao(["Este é um sistema especialista para fazer o diagnóstico ","sindrômico de abdomên agudo no recem nascido."])
meta("Diagn Sindrômico")
pergunta(1,"O RN apresenta sinais de desconforto respiratório?",["Sim","Não"],"",[],[],"Desconforto Respiratório")
pergunta(2,"Como se apresentava o abdômen do paciente?",["Sem distensão abdominal","Distensão leve","Distensão de abdomên superior","Distensão de todo o abdomên"],"",[],[],"Distensão Abdominal")
pergunta(3,"Quanto a eliminação de mecônio:",["Não houve","presente; em pequena quantidade e cinzenta","presente após estimulação","presente e normal (nas 1as 24 h)"],"",[],[],"Eliminação de mecônio")
pergunta(4,"Existe icterícia?",["Sim","Não"],"",[],[],"Icterícia")
pergunta(5,"O que se encontrou no toque retal?",["Sem anormalidades","Acompanhado de dor","Seguido de eliminação de fezes explosivas","Mal formação anorretal","Acompanhado de sangue"],"",[],[],"Toque retal")
pergunta(6,"Como eram os vômitos?",["Ausentes","Ocasionais","Precoces","Tardios"],"",[],[],"Vômitos")
regra(1,"Sim","Diagn Sindrômico","Síndrome Hemorrágica",0.5,["Sim","Vômitos","Ocasionais"])
regra(2,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.5,["Sim","Vômitos","Ocasionais"])
regra(3,"Sim","Diagn Sindrômico","Síndrome Perfurativa",0.5,["Sim","Vômitos","Ocasionais"])
regra(4,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.2,["Sim","Vômitos","Ocasionais"])
regra(5,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.7,["Sim","Vômitos","Precoces"])
regra(6,"Sim","Diagn Sindrômico","Síndrome Perfurativa",0.3,["Sim","Vômitos","Precoces"])
regra(7,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.4,["Sim","Vômitos","Precoces"])
regra(8,"Sim","Diagn Sindrômico","Síndrome Perfurativa",0.5,["Sim","Vômitos","Tardios"])
regra(9,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.5,["Sim","Vômitos","Tardios"])
regra(10,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.8,["Sim","Vômitos","Tardios"])
regra(11,"Sim","Diagn Sindrômico","Síndrome Hemorrágica",0.5,["Sim","Distensão Abdominal","Distensão leve"])
regra(12,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.5,["Sim","Distensão Abdominal","Distensão leve"])
regra(13,"Sim","Diagn Sindrômico","Síndrome Perfurativa",0.5,["Sim","Distensão Abdominal","Distensão leve"])
regra(14,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.3,["Sim","Distensão Abdominal","Distensão leve"])
regra(15,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.2,["Sim","Distensão Abdominal","Distensão de abdomên superior"])
regra(16,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.5,["Sim","Distensão Abdominal","Distensão de abdomên superior"])
regra(17,"Sim","Diagn Sindrômico","Síndrome Hemorrágica",0.3,["Sim","Distensão Abdominal","Distensão de todo o abdomên"])
regra(18,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.5,["Sim","Distensão Abdominal","Distensão de todo o abdomên"])
regra(19,"Sim","Diagn Sindrômico","Síndrome Perfurativa",0.5,["Sim","Distensão Abdominal","Distensão de todo o abdomên"])
regra(20,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.7,["Sim","Distensão Abdominal","Distensão de todo o abdomên"])
regra(21,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.2,["Sim","Desconforto Respiratório","Sim"])
regra(22,"Sim","Diagn Sindrômico","Síndrome Perfurativa",0.3,["Sim","Desconforto Respiratório","Sim"])
```

```
regra(23,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.5,["Sim","Desconforto Respiratório","Sim"])
regra(24,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.9,["Sim","Eliminação de mecônio","Não houve"])
regra(25,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.8,["Sim","Eliminação de mecônio","presente; em pequena quantidade e cinzenta"])
regra(26,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.6,["Sim","Eliminação de mecônio","presente após estimulação"])
regra(27,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.3,["Sim","Eliminação de mecônio","presente após estimulação"])
regra(28,"Sim","Diagn Sindrômico","Síndrome Hemorrágica",0.2,["Sim","Icterícia","Sim"])
regra(29,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.4,["Sim","Icterícia","Sim"])
regra(30,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.3,["Sim","Icterícia","Sim"])
regra(31,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.6,["Sim","Toque retal","Acompanhado de dor"])
regra(32,"Sim","Diagn Sindrômico","Síndrome Perfurativa",0.7,["Sim","Toque retal","Acompanhado de dor"])
regra(33,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.3,["Sim","Toque retal","Seguido de eliminação de fezes explosivas"])
regra(34,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.7,["Sim","Toque retal","Seguido de eliminação de fezes explosivas"])
regra(35,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.95,["Sim","Toque retal","Mal formação anorretal"])
regra(36,"Sim","Diagn Sindrômico","Síndrome Hemorrágica",0.2,["Sim","Toque retal","Acompanhado de sangue"])
regra(37,"Sim","Diagn Sindrômico","Síndrome Inflamatória",0.6,["Sim","Toque retal","Acompanhado de sangue"])
regra(38,"Sim","Diagn Sindrômico","Síndrome Obstrutiva",0.4,["Sim","Toque retal","Acompanhado de sangue"])
explicacao(1,[""])
explicacao(2,[""])
explicacao(3,[""])
explicacao(4,[""])
explicacao(5,[""])
explicacao(6,[""])
historia([])
```

ANEXO 5 – BASE DE CONHECIMENTO II – DIAGNÓSTICO DE ABDÔMEN AGUDO

SOBRE O SISTEMA ESPECIALISTA
-- Nome: Abdomen Agudo no Recém-Nascido
-- Autores: Angevaldo Lima

SE Vômitos = ocasionais
ENTÃO Diagnóstico Sindrômico = Síndrome Hemorrágica CNF 50%
Diagnóstico Sindrômico = Síndrome Inflamatória CNF 50%
Diagnóstico Sindrômico = Síndrome Perfurativa CNF 50%
Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 20%
SE Vômitos = precoces
ENTÃO Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 70%
Diagnóstico Sindrômico = Síndrome Inflamatória CNF 40%
Diagnóstico Sindrômico = Síndrome Perfurativa CNF 30%
SE Vômitos = tardios
ENTÃO Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 80%
Diagnóstico Sindrômico = Síndrome Inflamatória CNF 50%
Diagnóstico Sindrômico = Síndrome Perfurativa CNF 50%
SE Distensão Abdominal = distensão leve
ENTÃO Diagnóstico Sindrômico = Síndrome Hemorrágica CNF 50%
Diagnóstico Sindrômico = Síndrome Inflamatória CNF 50%
Diagnóstico Sindrômico = Síndrome Perfurativa CNF 50%
Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 30%
SE Distensão Abdominal = distensão de abdômen superior
ENTÃO Diagnóstico Sindrômico = Síndrome Inflamatória CNF 20%
Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 50%
SE Distensão Abdominal = distensão de todo abdômen
ENTÃO Diagnóstico Sindrômico = Síndrome Hemorrágica CNF 30%
Diagnóstico Sindrômico = Síndrome Inflamatória CNF 50%
Diagnóstico Sindrômico = Síndrome Perfurativa CNF 50%
Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 70%
SE Desconforto Respiratório = Sim
ENTÃO Diagnóstico Sindrômico = Síndrome Inflamatória CNF 20%
Diagnóstico Sindrômico = Síndrome Perfurativa CNF 30%
Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 50%
SE Eliminação de Mecônio = não houve
ENTÃO Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 90%
SE Eliminação de Mecônio = presente; em pequena quantidade e cinzenta
ENTÃO Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 80%
SE Eliminação de Mecônio = presente; após estimulação
ENTÃO Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 60%
Diagnóstico Sindrômico = Síndrome Inflamatória CNF 30%
SE Icterícia = Sim
ENTÃO Sindrômico = Síndrome Hemorrágica CNF 20%
Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 40%
Diagnóstico Sindrômico = Síndrome Inflamatória CNF 30%
SE Toque Retal = acompanhado de dor
ENTÃO Diagnóstico Sindrômico = Síndrome Inflamatória CNF 60%
Diagnóstico Sindrômico = Síndrome Perfurativa CNF 70%

SE Toque Retal = seguido da eliminação de fezes explosivas
ENTÃO Diagnóstico Sindrômico = Síndrome Inflamatória CNF 30%
Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 70%
SE Toque Retal = mal formação anorretal
ENTÃO Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 95%
SE Toque Retal = acompanhado de sangue
ENTÃO Diagnóstico Sindrômico = Síndrome Hemorrágica CNF 20%
Diagnóstico Sindrômico = Síndrome Inflamatória CNF 60%
Diagnóstico Sindrômico = Síndrome Obstrutiva CNF 40%

ANEXO 6 – RESULTADOS DA BASE DE CONHECIMENTO VINHOS

	SSEF		Expert SINTA	
Variáveis	Resposta	Valor de Verdade	Resposta	Fator Certeza
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Branco	Chablis Sauvignon Blanc Chardonay Soave	0,971 0,934 0,138 0,108	Chablis Sauvignon Blanc	71,517 60,225
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,849 0,772 0,138 0,108	Chablis Sauvignon Blanc	50,902 42,865
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Branco	Chablis Sauvignon Blanc Chardonay Soave	0,820 0,742 0,138 0,108	Chablis Sauvignon Blanc	47,837 40,264
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,653 0,570 0,138 0,108	Chablis Sauvignon Blanc	34,048 28,672
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Branco	Chablis Chardonay Sauvignon Blanc Soave	0,820 0,753 0,742 0,630	Chablis Chardonay Sauvignon Blanc Soave	47,837 40,788 40,284 31,724
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Tinto	Chablis Chardonay Sauvignon Blanc Soave	0,653 0,570 0,548 0,434	Chablis Chardonay Sauvignon Blanc Soave	34,048 29,03 28,672 22,579
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Branco	Chablis Sauvignon Blanc Riesling Chardonay Gamay Pinot Noir Chenin Blanc Zinfandel Soave Cab. Sauvignon	0,917 0,861 0,313 0,300 0,283 0,283 0,283 0,269 0,240 0,223	Chablis Sauvignon Blanc	60,484 50,934
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Tinto	Chablis Sauvignon Blanc Riesling Chardonay Gamay Pinot Noir Chenin Blanc Zinfandel Soave Cab. Sauvignon	0,764 0,685 0,313 0,300 0,283 0,283 0,283 0,269 0,240 0,223	Chablis Sauvignon Blanc	43,049 36,252

Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Branco	Chablis Sauvignon Blanc Riesling Chardonay Gamay Pinot Noir Chenin Blanc Zinfandel Soave Cab. Sauvignon	0,715 0,637 0,313 0,300 0,283 0,283 0,283 0,269 0,240 0,223	Chablis Sauvignon Blanc	36,903 31,076
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Tinto	Chablis Sauvignon Blanc Riesling Chardonay Gamay Pinot Noir Chenin Blanc Zinfandel Soave Cab. Sauvignon	0,562 0,490 0,313 0,300 0,283 0,283 0,283 0,269 0,240 0,223	Chablis Sauvignon Blanc	26,266 22,118
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Branco	Chablis Chardonay Sauvignon Blanc Soave Riesling Gamay Pinot Noir Chenin Blanc Zinfandel Cab. Sauvignon	0,715 0,696 0,637 0,585 0,313 0,283 0,283 0,283 0,269 0,223	Chablis Chardonay Sauvignon Blanc Soave	36,903 31,465 31,076 24,473
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Tinto	Chablis Chardonay Sauvignon Blanc Soave Riesling Gamay Pinot Noir Chenin Blanc Zinfandel Cab. Sauvignon	0,562 0,547 0,490 0,447 0,313 0,283 0,283 0,283 0,269 0,223	Chablis Chardonay Sauvignon Blanc Soave	26,266 22,395 22,118 17,418
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Branco	Riesling Chardonay Chenin Blanc Gamay Soave Pinot Noir Zinfandel Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,634 0,612 0,545 0,528 0,507 0,485 0,190 0,171 0,161	Riesling Chardonay Chenin Blanc Soave Gamay Pinot Noir Zinfandel	29,134 26,261 25,897 20,394 18,662 16,796 15,863

Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Riesling Chardonay Chenin Blanc Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,634 0,609 0,545 0,507 0,499 0,414 0,190 0,171 0,161	Gamay Pinot Noir Zinfandel Riesling Chardonay Chenin Blanc Soave	27,564 24,808 23,43 20,736 18,662 18,432 14,515
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Branco	Riesling Chenin Blanc Gamay Pinot Noir Chardonay Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,612 0,545 0,310 0,310 0,295 0,249 0,190 0,171 0,161	Riesling Chenin Blanc Gamay	29,134 25,897 18,662
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Tinto	Gamay Riesling Chenin Blanc Pinot Noir Chardonay Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,545 0,499 0,310 0,310 0,295 0,249 0,190 0,171 0,161	Gamay Riesling Chenin Blanc	27,564 20,736 18,432
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Branco	Riesling Chenin Blanc Gamay Pinot Noir Chardonay Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,612 0,545 0,310 0,310 0,295 0,249 0,190 0,171 0,161	Riesling Chenin Blanc Gamay	50,111 44,543 32,099
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Riesling Chenin Blanc Pinot Noir Chardonay Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,545 0,499 0,310 0,310 0,295 0,249 0,190 0,171 0,161	Gamay Riesling Chenin Blanc	47,411 35,666 31,703

Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Branco	Riesling Chardonay Chenin Blanc Soave Gamay Pinot Noir Zinfandel Chablis Cab. Sauvignon Sauvignon Blanc	0,864 0,849 0,813 0,733 0,289 0,255 0,243 0,190 0,171 0,161	Riesling Chardonay Chenin Blanc Soave	52,441 47,197 46,615 36,709
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Tinto	Riesling Chardonay Chenin Blanc Soave Gamay Pinot Noir Zinfandel Chablis Cab. Sauvignon Sauvignon Blanc	0,703 0,671 0,643 0,547 0,289 0,255 0,243 0,190 0,171 0,161	Riesling Chardonay Chenin Blanc Soave	37,325 33,592 33,178 26,127
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Branco	Riesling Chenin Blanc Gamay Chardonay Pinot Noir Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,864 0,813 0,289 0,267 0,255 0,243 0,213 0,190 0,171 0,161	Riesling Chenin Blanc	52,441 46,615
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Tinto	Riesling Chenin Blanc Gamay Chardonay Pinot Noir Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,703 0,643 0,289 0,267 0,255 0,243 0,213 0,190 0,171 0,161	Riesling Chenin Blanc	37,325 33,178
Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Branco	Riesling Chenin Blanc Gamay Chardonay Pinot Noir Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,980 0,961 0,289 0,267 0,255 0,243 0,213 0,190 0,171 0,161	Riesling Chenin Blanc	73,208 65,074

Prato Principal = Carne; Tem Vitela = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Tinto	Riesling Chenin Blanc Gamay Chardonay Pinot Noir Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,861 0,811 0,289 0,267 0,255 0,243 0,213 0,190 0,171 0,161	Riesling Chenin Blanc	52,105 46,316
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Branco	Cab. Sauvignon Pinot Noir Zinfandel	0,822 0,138 0,131	Cab. Sauvignon	51,237
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Tinto	Cab. Sauvignon Pinot Noir Zinfandel	0,954 0,138 0,131	Cab. Sauvignon	67,376
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Branco	Cab. Sauvignon Pinot Noir Zinfandel	0,598 0,138 0,131	Cab. Sauvignon	34,272
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Tinto	Cab. Sauvignon Pinot Noir Zinfandel	0,770 0,138 0,131	Cab. Sauvignon	45,068
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Branco	Cab. Sauvignon Pinot Noir Zinfandel	0,598 0,548 0,520	Cab. Sauvignon Pinot Noir Zinfandel	34,272 32,659 30,845
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Tinto	Cab. Sauvignon Pinot Noir Zinfandel	0,770 0,753 0,725	Cab. Sauvignon Pinot Noir Zinfandel	45,068 42,947 40,561
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Branco	Cab. Sauvignon Gamay Pinot Noir Zinfandel	0,743 0,412 0,400 0,382	Cab. Sauvignon	43,333
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Tinto	Cab. Sauvignon Gamay Pinot Noir Zinfandel	0,895 0,412 0,400 0,382	Cab. Sauvignon	56,982
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Branco	Cab. Sauvignon Gamay Pinot Noir Zinfandel	0,566 0,412 0,400 0,382	Cab. Sauvignon	26,438
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Tinto	Cab. Sauvignon Gamay Pinot Noir Zinfandel	0,700 0,412 0,400 0,382	Cab. Sauvignon	34,766
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Branco	Pinot Noir Zinfandel Cab. Sauvignon Gamay	0,612 0,588 0,566 0,412	Cab. Sauvignon Pinot Noir Zinfandel	26,438 25,194 23,795
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Tinto	Pinot Noir Zinfandel Cab. Sauvignon Gamay	0,740 0,715 0,700 0,412	Cab. Sauvignon Pinot Noir Zinfandel	34,766 33,13 31,29

Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,736 0,702 0,678 0,245	Gamay Pinot Noir Zinfandel	25,194 22,675 21,415
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,803 0,778 0,755 0,245	Gamay Pinot Noir Zinfandel	31,498 28,348 26,773
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,736 0,427 0,408 0,245	Gamay	25,194
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,803 0,427 0,408 0,245	Gamay	31,498
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,736 0,427 0,408 0,245	Gamay	43,334
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,803 0,427 0,408 0,245	Gamay	54,176
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,750 0,719 0,690 0,245	Gamay Pinot Noir Zinfandel	41,99 37,791 35,692
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,885 0,871 0,848 0,245	Gamay Pinot Noir Zinfandel	55,217 49,696 46,935
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,750 0,363 0,346 0,245	Gamay	41,99
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,885 0,363 0,346 0,245	Gamay	55,217
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,903 0,363 0,346 0,245	Gamay	58,619
Prato Principal = Carne; Tem Vitela = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,986 0,363 0,346 0,245	Gamay	77,083
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Branco	Cab. Sauvignon Pinot Noir Zinfandel	0,799 0,138 0,131	Cab. Sauvignon	45,544
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Tinto	Cab. Sauvignon Pinot Noir Zinfandel	0,949 0,138 0,131	Cab. Sauvignon	63,989

Prato Principal = Ave; Tem Peru = Sim Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Branco	Cab. Sauvignon Pinot Noir Zinfandel	0,598 0,138 0,131	Cab. Sauvignon	30,464
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Tinto	Cab. Sauvignon Pinot Noir Zinfandel	0,770 0,138 0,131	Cab. Sauvignon	42,802
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Branco	Cab. Sauvignon Pinot Noir Zinfandel	0,598 0,548 0,520	Cab. Sauvignon Pinot Noir Zinfandel	30,464 29,03 27,418
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Tinto	Cab. Sauvignon Pinot Noir Zinfandel	0,770 0,753 0,725	Cab. Sauvignon Pinot Noir Zinfandel	42,802 40,788 38,522
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Branco	Cab. Sauvignon Gamay Pinot Noir Zinfandel	0,743 0,412 0,400 0,382	Cab. Sauvignon	38,518
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Tinto	Cab. Sauvignon Gamay Pinot Noir Zinfandel	0,895 0,412 0,400 0,382	Cab. Sauvignon	54,118
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Branco	Cab. Sauvignon Gamay Pinot Noir Zinfandel	0,566 0,412 0,400 0,382	Cab. Sauvignon	23,501
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Tinto	Cab. Sauvignon Gamay Pinot Noir Zinfandel	0,700 0,412 0,400 0,382	Cab. Sauvignon	33,019
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Branco	Pinot Noir Zinfandel Cab. Sauvignon Gamay	0,612 0,588 0,566 0,412	Cab. Sauvignon Pinot Noir Zinfandel	23,501 22,395 21,151
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Tinto	Pinot Noir Zinfandel Cab. Sauvignon Gamay	0,740 0,715 0,700 0,412	Cab. Sauvignon Pinot Noir Zinfandel	33,019 31,465 29,717
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,736 0,702 0,678 0,245	Gamay Pinot Noir Zinfandel	24,468 22,022 20,798
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,803 0,778 0,755 0,245	Gamay Pinot Noir Zinfandel	31,207 28,086 26,526
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,736 0,427 0,408 0,245	Gamay	24,468
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,803 0,427 0,408 0,245	Gamay	31,207

Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,736 0,427 0,408 0,245	Gamay	42,086
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,803 0,427 0,408 0,245	Gamay	53,676
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,745 0,714 0,686 0,245	Gamay Pinot Noir Zinfandel	37,325 33,592 31,726
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,883 0,869 0,845 0,245	Gamay Pinot Noir Zinfandel	52,441 47,197 44,575
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,745 0,363 0,346 0,245	Gamay	37,325
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,883 0,363 0,346 0,245	Gamay	52,441
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,881 0,363 0,346 0,245	Gamay	52,105
Prato Principal = Ave; Tem Peru = Sim Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon	0,983 0,363 0,346 0,245	Gamay	73,208
Prato Principal = Ave; Tem Peru = Não Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Branco	Chablis Sauvignon Blanc Chardonay Soave	0,975 0,941 0,138 0,108	Chablis Sauvignon Blanc	75,303 63,413
Prato Principal = Ave; Tem Peru = Não Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,871 0,794 0,138 0,108	Chablis Sauvignon Blanc	57,264 48,223
Prato Principal = Ave; Tem Peru = Não Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Branco	Chablis Sauvignon Blanc Chardonay Soave	0,820 0,742 0,138 0,108	Chablis Sauvignon Blanc	50,37 42,417
Prato Principal = Ave; Tem Peru = Não Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,653 0,570 0,138 0,108	Chablis Sauvignon Blanc	38,304 32,256
Prato Principal = Ave; Tem Peru = Não Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Branco	Chablis Chardonay Sauvignon Blanc Soave	0,820 0,753 0,742 0,630	Chablis Chardonay Sauvignon Blanc Soave	50,37 42,947 42,417 33,403
Prato Principal = Ave; Tem Peru = Não Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,653 0,570 0,548 0,434	Chablis Chardonay Sauvignon Blanc Soave	38,304 32,659 32,256 25,402

Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Branco	Chablis Sauvignon Blanc Riesling Chardonay Pinot Noir Gamay Chenin Blanc Zinfandel Soave Cab. Sauvignon	0,917 0,861 0,313 0,300 0,283 0,283 0,283 0,269 0,240 0,223	Chablis Sauvignon Blanc	63,686 53,63
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Tinto	Chablis Sauvignon Blanc Riesling Chardonay Pinot Noir Gamay Chenin Blanc Zinfandel Soave Cab. Sauvignon	0,764 0,685 0,313 0,300 0,283 0,283 0,283 0,269 0,240 0,223	Chablis Sauvignon Blanc	48,43 40,784
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Branco	Chablis Sauvignon Blanc Riesling Chardonay Pinot Noir Gamay Chenin Blanc Zinfandel Soave Cab. Sauvignon	0,715 0,637 0,313 0,300 0,283 0,283 0,283 0,269 0,240 0,223	Chablis Sauvignon Blanc	38,857 32,721
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Tinto	Chablis Sauvignon Blanc Riesling Chardonay Pinot Noir Gamay Chenin Blanc Zinfandel Soave Cab. Sauvignon	0,562 0,490 0,313 0,300 0,283 0,283 0,283 0,269 0,240 0,223	Chablis Sauvignon Blanc	29,549 24,883
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Branco	Chablis Chardonay Sauvignon Blanc Soave Riesling Pinot Noir Gamay Chenin Blanc Zinfandel Cab. Sauvignon	0,715 0,696 0,637 0,585 0,313 0,283 0,283 0,283 0,269 0,223	Chablis Chardonay Sauvignon Blanc Soave	38,857 33,13 32,721 25,768

Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Tinto	Chablis Chardonay Sauvignon Blanc Soave Riesling Pinot Noir Gamay Chenin Blanc Zinfandel Cab. Sauvignon	0,562 0,547 0,490 0,447 0,313 0,283 0,283 0,283 0,269 0,223	Chablis Chardonay Sauvignon Blanc Soave	29,549 25,194 24,883 19,596
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Branco	Riesling Chardonay Chenin Blanc Gamay Soave Pinot Noir Zinfandel Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,634 0,612 0,545 0,528 0,507 0,485 0,190 0,171 0,161	Riesling Chardonay Chenin Blanc Soave Gamay Pinot Noir Zinfandel	30,676 27,609 27,268 21,473 18,662 16,796 15,863
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Riesling Chardonay Chenin Blanc Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,634 0,609 0,545 0,507 0,499 0,414 0,190 0,171 0,161	Gamay Pinot Noir Zinfandel Riesling Chardonay Chenin Blanc Soave	27,564 24,808 23,43 23,328 20,995 20,736 16,33
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Branco	Riesling Chenin Blanc Gamay Pinot Noir Chardonay Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,612 0,545 0,310 0,310 0,295 0,249 0,190 0,171 0,161	Riesling Chenin Blanc Gamay	30,676 27,268 18,662
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Tinto	Gamay Riesling Chenin Blanc Pinot Noir Chardonay Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,545 0,499 0,310 0,310 0,295 0,249 0,190 0,171 0,161	Gamay Riesling Chenin Blanc	27,564 23,328 20,736

Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Branco	Riesling Chenin Blanc Gamay Pinot Noir Chardonay Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,612 0,545 0,310 0,310 0,295 0,249 0,190 0,171 0,161	Riesling Chenin Blanc Gamay	52,763 46,901 32,099
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Riesling Chenin Blanc Pinot Noir Chardonay Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,662 0,545 0,499 0,310 0,310 0,295 0,249 0,190 0,171 0,161	Gamay Riesling Chenin Blanc	47,741 40,124 35,666
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Branco	Riesling Chardonay Chenin Blanc Soave Gamay Pinot Noir Zinfandel Chablis Cab. Sauvignon Sauvignon Blanc	0,866 0,852 0,816 0,736 0,289 0,255 0,243 0,190 0,171 0,161	Riesling Chardonay Chenin Blanc Soave	55,217 49,696 49,082 38,652
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Tinto	Riesling Chardonay Chenin Blanc Soave Gamay Pinot Noir Zinfandel Chablis Cab. Sauvignon Sauvignon Blanc	0,708 0,677 0,648 0,551 0,289 0,255 0,243 0,190 0,171 0,161	Riesling Chardonay Chenin Blanc Soave	41,99 37,791 37,325 29,393
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Branco	Riesling Chenin Blanc Gamay Chardonay Pinot Noir Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,866 0,816 0,289 0,267 0,255 0,243 0,213 0,190 0,171 0,161	Riesling Chenin Blanc	55,221 7 49,082

Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Tinto	Riesling Chenin Blanc Gamay Chardonay Pinot Noir Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,708 0,648 0,289 0,267 0,255 0,243 0,213 0,190 0,171 0,161	Riesling Chenin Blanc	41,99 37,325
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Branco	Riesling Chenin Blanc Gamay Chardonay Pinot Noir Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,984 0,967 0,289 0,267 0,255 0,243 0,213 0,190 0,171 0,161	Riesling Chenin Blanc	77,083 68,519
Prato Principal = Ave; Tem Peru = Não Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Tinto	Riesling Chenin Blanc Gamay Chardonay Pinot Noir Zinfandel Soave Chablis Cab. Sauvignon Sauvignon Blanc	0,887 0,838 0,289 0,267 0,255 0,243 0,213 0,190 0,171 0,161	Riesling Chenin Blanc	52,105 40,526
Prato Principal = Vegetariano Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Branco	Chablis Sauvignon Blanc Cab. Sauvignon Pinot Noir Chardonay Zinfandel Soave Riesling Gamay Chenin Blanc	0,937 0,879 0,233 0,130 0,130 0,123 0,102 0,063 0,063 0,056	Chablis Sauvignon Blanc	80,35 67,663
Prato Principal = Vegetariano Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Tinto	Cab. Sauvignon Chablis Sauvignon Blanc Pinot Noir Chardonay Zinfandel Soave Riesling Gamay Chenin Blanc	0,901 0,258 0,221 0,130 0,130 0,123 0,102 0,063 0,063 0,056	Cab. Souvignon	71,892

Prato Principal = Vegetariano Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Branco	Chablis Chardonay Sauvignon Blanc Soave Cab. Sauvignon Pinot Noir Zinfandel Riesling Gamay Chenin Blanc	0,604 0,598 0,520 0,472 0,212 0,155 0,147 0,063 0,063 0,056	Chardonay Chablis Sauvignon Blanc Soave	51,03 47,88 40,32 39,69
Prato Principal = Vegetariano Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Tinto	Pinot Noir Zinfandel Cab. Sauvignon Chablis Sauvignon Blanc Chardonay Soave Riesling Gamay Chenin Blanc	0,598 0,567 0,549 0,235 0,200 0,155 0,122 0,063 0,063 0,056	Pinot Noir Zinfandel Cab. Souvignon	51,03 48,195 42,84
Prato Principal = Vegetariano Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Branco	Chablis Riesling Chardonay Sauvignon Blanc Chenin Blanc Soave Cab. Sauvignon Pinot Noir Zinfandel Gamay	0,604 0,567 0,535 0,520 0,504 0,421 0,212 0,147 0,139 0,09	Riesling Chablis Chenin Blanc Chardonay Sauvignon Blanc Soave	51,03 47,88 45,36 40,824 40,32 31,752
Prato Principal = Vegetariano Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Cab. Sauvignon Pinot Noir Zinfandel Chablis Sauvignon Blanc Chardonay Soave Riesling Chenin Blanc	0,567 0,549 0,535 0,507 0,235 0,200 0,147 0,115 0,09 0,08	Gamay Cab. Souvignon Pinot Noir Zinfandel	51,03 42,84 40,824 38,556
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Branco	Chablis Sauvignon Blanc Gamay Pinot Noir Zinfandel Cab. Sauvignon Riesling Chardonay Chenin Blanc Soave	0,884 0,815 0,508 0,504 0,483 0,475 0,223 0,216 0,200 0,171	Chablis Sauvignon Blanc	76,264 64,222

Prato Principal = Vegetariano Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Tinto	Cab. Sauvignon Gamay Pinot Noir Zinfandel Chablis Riesling Sauvignon Blanc Chardonay Chenin Blanc Soave	0,891 0,508 0,504 0,483 0,258 0,223 0,221 0,216 0,200 0,171	Cab. Souvignon	68,236
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Branco	Chardonay Pinot Noir Soave Gamay Zinfandel Chablis Cab. Sauvignon Sauvignon Blanc Riesling Chenin Blanc	0,637 0,523 0,512 0,508 0,502 0,501 0,456 0,431 0,223 0,200	Chardonay Soave Chablis Sauvignon Blanc	51,03 39,69 36,936 31,104
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Tinto	Pinot Noir Zinfandel Cab. Sauvignon Gamay Chardonay Chablis Riesling Sauvignon Blanc Chenin Blanc Soave	0,773 0,747 0,623 0,508 0,238 0,235 0,223 0,200 0,200 0,189	Pinot Noir Zinfandel Cab. Souvignon	51,03 48,195 33,048
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Branco	Riesling Chenin Blanc Gamay Pinot Noir Chablis Zinfandel Chardonay Cab. Sauvignon Sauvignon Blanc Soave	0,657 0,596 0,588 0,513 0,501 0,493 0,483 0,456 0,431 0,387	Riesling Chenin Blanc Chablis Chardonay Sauvignon Blanc Soave	51,03 45,36 36,936 31,493 31,104 24,494
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon Riesling Chenin Blanc Chablis Chardonay Sauvignon Blanc Soave	0,804 0,673 0,649 0,623 0,279 0,252 0,235 0,231 0,200 0,183	Gamay Cab. Souvignon Pinot Noir Zinfandel	51,03 33,048 31,493 29,743

Prato Principal = Vegetariano Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Branco	Gamay Cab. Sauvignon Pinot Noir Zinfandel Chablis Sauvignon Blanc Riesling Chenin Blanc Chardonay Soave	0,779 0,677 0,670 0,647 0,638 0,545 0,504 0,459 0,433 0,347	Chablis Sauvignon Blanc Cab. Sauvignon Riesling Chardonay Chenin Blanc Soave Gamay Pinot Noir Zinfandel	53,865 45,36 30,845 29,16 26,244 25,92 20,412 18,662 16,796 15,863
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Tinto	Cab. Sauvignon Gamay Pinot Noir Zinfandel Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,916 0,838 0,817 0,796 0,267 0,240 0,237 0,188 0,185 0,157	Cab. Sauvignon Gamay Pinot Noir Zinfandel	52,751 31,916 28,725 27,129
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Branco	Gamay Pinot Noir Zinfandel Chardonay Soave Riesling Chenin Blanc Cab. Sauvignon Chablis Sauvignon Blanc	0,779 0,771 0,747 0,640 0,515 0,504 0,459 0,372 0,159 0,135	Chardonay Soave Pinot Noir Zinfandel Riesling Chenin Blanc Gamay	51,03 39,69 32,659 30,845 29,16 25,92 18,662
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Tinto	Pinot Noir Zinfandel Gamay Cab. Sauvignon Riesling Chardonay Chenin Blanc Soave Chablis Sauvignon Blanc	0,948 0,934 0,838 0,372 0,267 0,244 0,240 0,194 0,159 0,135	Pinot Noir Zinfandel Gamay	55,854 52,751 31,916
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Branco	Gamay Riesling Chenin Blanc Pinot Noir Zinfandel Cab. Sauvignon Chardonay Soave Chablis Sauvignon Blanc	0,930 0,850 0,802 0,528 0,508 0,372 0,221 0,176 0,159 0,135	Riesling Chenin Blanc Gamay	69,587 61,855 44,536

Prato Principal = Vegetariano Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,982 0,528 0,508 0,372 0,288 0,259 0,221 0,176 0,159 0,135	Gamay	76,165
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Branco	Riesling Chardonay Chablis Chenin Blanc Gamay Sauvignon Blanc Soave Pinot Noir Zinfandel Cab. Sauvignon	0,672 0,639 0,638 0,611 0,591 0,545 0,511 0,483 0,463 0,391	Chablis Riesling Chardonay Chenin Blanc Sauvignon Blanc Soave	53,865 52,488 47,239 46,656 45,36 36,742
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,784 0,763 0,736 0,691 0,252 0,226 0,213 0,185 0,169 0,157	Gamay Cab. Souvignon Pinot Noir Zinfandel	52,488 48,195 47,239 44,615
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Branco	Riesling Chenin Blanc Chardonay Gamay Soave Pinot Noir Zinfandel Cab. Sauvignon Chablis Sauvignon Blanc	0,657 0,629 0,595 0,530 0,504 0,493 0,473 0,369 0,159 0,135	Riesling Chardonay Chenin Blanc Soave	52,488 51,03 46,656 39,69
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,784 0,758 0,732 0,369 0,252 0,226 0,221 0,175 0,159 0,135	Gamay Pinot Noir Zinfandel	52,488 51,03 48,195

Prato Principal = Vegetariano Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Branco	Riesling Chenin Blanc Gamay Pinot Noir Zinfandel Cab. Sauvignon Chardonay Chablis Soave Sauvignon Blanc	0,952 0,918 0,548 0,473 0,453 0,369 0,197 0,159 0,156 0,135	Riesling Chenin Blanc	78,141 69,459
Prato Principal = Vegetariano Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Tinto	Gamay Pinot Noir Zinfandel Cab. Sauvignon Riesling Chenin Blanc Chardonay Chablis Soave Sauvignon Blanc	0,970 0,473 0,453 0,369 0,273 0,246 0,197 0,159 0,156 0,135	Gamay	78,141
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Branco	Chablis Sauvignon Blanc Chardonay Soave	0,975 0,941 0,138 0,108	Chablis Sauvignon Blanc	75,303 63,413
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Seco Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,871 0,794 0,138 0,108	Chablis Sauvignon Blanc	57,264 48,223
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Branco	Chablis Sauvignon Blanc Chardonay Soave	0,820 0,742 0,138 0,108	Chablis Sauvignon Blanc	50,37 42,417
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Suave Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,653 0,570 0,138 0,108	Chablis Sauvignon Blanc	38,304 32,256
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Branco	Chablis Chardonay Sauvignon Blanc Soave	0,820 0,753 0,742 0,630	Chablis Chardonay Sauvignon Blanc Soave	50,37 42,947 42,417 33,403
Prato Principal = Peixe Tem Molho = Não Doçura Preferida = Doce Cor Preferida = Tinto	Chablis Sauvignon Blanc Chardonay Soave	0,653 0,570 0,548 0,434	Chablis Chardonay Sauvignon Blanc Soave	38,304 32,659 32,256 25,402
Prato Principal = Peixe Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Branco	Chablis Sauvignon Blanc Riesling Chardonay Chenin Blanc Soave	0,917 0,861 0,313 0,300 0,283 0,240	Chablis Sauvignon Blanc	63,686 53,63
Prato Principal = Peixe Tem Molho = Sim; Molho = Temperado Doçura Preferida = Seco Cor Preferida = Tinto	Chablis Sauvignon Blanc Riesling Chardonay Chenin Blanc Soave	0,764 0,685 0,313 0,300 0,283 0,240	Chablis Sauvignon Blanc	48,43 40,784

Prato Principal = Peixe Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Branco	Chablis Sauvignon Blanc Riesling Chardonay Chenin Blanc Soave	0,715 0,637 0,313 0,300 0,283 0,240	Chablis Sauvignon Blanc	38,857 32,721
Prato Principal = Peixe Tem Molho = Sim; Molho = Temperado Doçura Preferida = Suave Cor Preferida = Tinto	Chablis Sauvignon Blanc Riesling Chardonay Chenin Blanc Soave	0,562 0,490 0,313 0,300 0,283 0,240	Chablis Sauvignon Blanc	29,549 24,883
Prato Principal = Peixe Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Branco	Chablis Chardonay Sauvignon Blanc Soave Riesling Chenin Blanc	0,715 0,696 0,637 0,585 0,313 0,283	Chablis Chardonay Sauvignon Blanc Soave	38,857 33,13 32,721 25,768
Prato Principal = Peixe Tem Molho = Sim; Molho = Temperado Doçura Preferida = Doce Cor Preferida = Tinto	Chablis Chardonay Sauvignon Blanc Soave Riesling Chenin Blanc	0,562 0,547 0,490 0,447 0,313 0,283	Chablis Chardonay Sauvignon Blanc Soave	29,549 25,194 24,883 19,596
Prato Principal = Peixe Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Branco	Riesling Chardonay Chenin Blanc Soave Chablis Sauvignon Blanc	0,662 0,634 0,612 0,528 0,190 0,161	Riesling Chardonay Chenin Blanc Soave	30,676 27,609 27,268 21,473
Prato Principal = Peixe Tem Molho = Sim; Molho = Tomate Doçura Preferida = Seco Cor Preferida = Tinto	Riesling Chardonay Chenin Blanc Soave Chablis Sauvignon Blanc	0,545 0,507 0,499 0,414 0,190 0,161	Riesling Chardonay Chenin Blanc Soave	23,328 20,995 20,736 16,33
Prato Principal = Peixe Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Branco	Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,662 0,612 0,310 0,249 0,190 0,161	Riesling Chenin Blanc	30,676 27,268
Prato Principal = Peixe Tem Molho = Sim; Molho = Tomate Doçura Preferida = Suave Cor Preferida = Tinto	Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,545 0,499 0,310 0,249 0,190 0,161	Riesling Chenin Blanc	23,328 20,736
Prato Principal = Peixe Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Branco	Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,662 0,612 0,310 0,249 0,190 0,161	Riesling Chenin Blanc	52,763 46,901

Prato Principal = Peixe Tem Molho = Sim; Molho = Tomate Doçura Preferida = Doce Cor Preferida = Tinto	Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,545 0,499 0,310 0,249 0,190 0,161	Riesling Chenin Blanc	40,124 35,666
Prato Principal = Peixe Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Branco	Riesling Chardonay Chenin Blanc Soave Chablis Sauvignon Blanc	0,866 0,852 0,816 0,736 0,190 0,161	Riesling Chardonay Chenin Blanc Soave	55,217 49,696 49,082 38,652
Prato Principal = Peixe Tem Molho = Sim; Molho = Doce Doçura Preferida = Seco Cor Preferida = Tinto	Riesling Chardonay Chenin Blanc Soave Chablis Sauvignon Blanc	0,708 0,677 0,648 0,551 0,190 0,161	Riesling Chardonay Chenin Blanc Soave	41,99 37,791 37,325 29,393
Prato Principal = Peixe Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Branco	Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,866 0,816 0,267 0,213 0,190 0,161	Riesling Chenin Blanc	55,217 49,082
Prato Principal = Peixe Tem Molho = Sim; Molho = Doce Doçura Preferida = Suave Cor Preferida = Tinto	Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,708 0,648 0,267 0,213 0,190 0,161	Riesling Chenin Blanc	41,99 37,325
Prato Principal = Peixe Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Branco	Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,984 0,967 0,267 0,213 0,190 0,161	Riesling Chenin Blanc	77,083 68,519
Prato Principal = Peixe Tem Molho = Sim; Molho = Doce Doçura Preferida = Doce Cor Preferida = Tinto	Riesling Chenin Blanc Chardonay Soave Chablis Sauvignon Blanc	0,887 0,838 0,267 0,213 0,190 0,161	Riesling Chenin Blanc	58,619 52,105

REFERÊNCIAS BIBLIOGRÁFICAS

ADLASSNIG, K. P.; 1986. Fuzzy Set Theory in Medical Diagnosis. *IEEE Transactions on System, Man, and Cybernetics*, v. SMC-16, n. 2 (Mar/Apr), pg 260-265.

ALGARVE, A. S.; DE 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.; 1989. *Inteligência Artificial: Um curso prático*. Rio de Janeiro: Livros Técnicos e Científicos Editora Ltda.

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

BRASIL, L. M.; 1999. *Proposta de Arquitetura para Sistema Especialista Híbrido e a Correspondente Metodologia de Aquisição de Conhecimento*. Florianópolis. Tese (Doutorado em Engenharia Elétrica) – Centro Tecnológico, Universidade Federal de Santa Catarina.

CLOCKSIN, W. F.; MELLISH, C.S.; 1987. *Programming in prolog*. Third Revised and Extended Edition. USA: Springer-Verlag.

DE AZEVEDO, F. M.; 1993. *Contribution to the Study of Neural Networks in Dinamical Expert Systems*. Namur, Belgium. Thesis (Doctor of Science) – Facultés Universitaires Notre Dame de La Paix.

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

GERSTING, J. L.; 1995. *Fundamentos Matemáticos para a Ciência da Computação*. 3. ed. Rio de Janeiro: Livros Técnicos e Científicos Editora.

LIMA, A.; 2000. *Sistema Especialista para Diagnóstico de Abdômen Agudo*. Exercício desenvolvido na disciplina "Engenharia Biomédica e Informática Médica para Medicina", Curso de Pós-graduação em Ciências Médicas, UFSC.

LOPES, H. S.; 1996. *Analogia e Aprendizado Evolucionário: Aplicação em Diagnóstico Clínico*. Florianópolis. Tese (Doutorado em Engenharia Elétrica) – Centro Tecnológico, Universidade Federal de Santa Catarina.

MINSKI, M. L.; PAPERT, S. A.; 1988. *Perceptrons: An Introduction to Computational Geometry*. 3. ed. The MIT Press.

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.

MONTELLO, M. V.; OKAMOTO, R. M.; PELLEGRINE, G. F.; ZIMMERMANN, A. C.; 1997. *Visual Prolog Expert System*. Exercício desenvolvido na disciplina "Inteligência Artificial", Curso de Pós-Graduação em Engenharia Elétrica, UFSC.

NASSAR, S. M.; 1998. *Informática e Estatística: Uma Interação entre duas Ciências*. Florianópolis. Trabalho Submetido ao Concurso de Professor Titular – Departamento de Informática e Estatística, UFSC.

NOLT, J.; ROHATYN, D.; 1991. *Lógica*. Sao Paulo: McGraw-Hill.

RABUSKE, R. A.; 1995. *Inteligência Artificial*. Florianópolis: Editora da UFSC.

RICH, E.; 1988. *Inteligência Artificial*. São Paulo: Makron Books do Brasil Editora.

RICH, E.; KNIGHT, K.; 1994. *Inteligência Artificial*. São Paulo: Makron Books do Brasil Editora.

ROBINSON, J.A.; 1965. *A machine Oriented Logic based on the Resolution Principle*. J. ACM, 12(1). p. 23-41

ROBINSON, P. R.; 1988. *Turbo Prolog – Guia do Usuário.* São Paulo: Editora McGraw-Hill.

RODDER, W.; MEYER, K. H.; 1996. *Sistemas Especialistas Probabilísticos: Uma análise da Shell Spirit-95.* 16^0 ENEGEP, Piracicaba.

SANDRI, S.; CORREA, C.; 1999. Lógica Nebulosa. In: V ESCOLA DE REDES NEURAIS (19.: Julho 1998: São José dos Campos, São Paulo). Anais. São Paulo, 1999. p c073-c090.

TURBAN, E.; 1992. *Expert Systems and Applied Artificial Inteliggence.* USA: Ed. Moura.

WIDMAN, L. E.; 1998. Sistemas Especialistas em Medicina. *Informática Médica*, v. 1, n. 5 (Set/Out).