UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE PÓS GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

SISTEMA ESPECIALISTA DIFUSO APLICADO AO PROCESSO DE ANÁLISE QUÍMICA QUALITATIVA DE AMOSTRAS DE MINERAIS

por

Anita Maria da Rocha Fernandes

Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do grau de Mestre em Ciência da Computação

Prof. Rogério Cid Bastos, Dr.
Orientador

Florianópolis, Fevereiro de 1996

SISTEMA ESPECIALISTA DIFUSO APLICADO AO PROCESSO DE ANÁLISE QUÍMICA QUALITATIVA DE AMOSTRAS DE MINERAIS

ANITA MARIA DA ROCHA FERNANDES

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO TÍTULO DE

MESTRE EM CIÊNCIA DA COMPUTAÇÃO

ESPECIALIDADE SISTEMAS DE CONHECIMENTO E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS - GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO.

Prof. Rogério Cid Bastos, Dr.
Coordenador do Curso

BANCA EXAMINADORA:

Prof. Rogério Cid Bastos, Dr.
Orientador

Prof. Luiz Fernando Jacintho Maia, Dr.

Prof. César Augusto Mortari, Dr.

Prof. Roberto Pacheco, M.Sc.

Aos meus queridos pais Luiz e Dilma,
com todo o meu carinho e gratidão.

AGRADECIMENTOS

Agradeço primeiramente a Deus e a minha família - meus pais Luiz e Dilma, minhas irmãs Ana, Betinha e Lu; a vó Anita e a tia Beth, pela paciência, carinho, incentivo e compreensão.

Ao meu **orientador** Rogério pelos ensinamentos, pela paciência e por me fazer acreditar em minha capacidade. Ao meu **amigo** Rogério, pela amizade, pela compreensão, pelo empenho em tentar me mostrar que as coisas podem ser diferentes, basta acreditar.

A Angelita pelo apoio, amizade e incentivo nestes dois anos de caminhada. Ao Cesar e a Valeria pela amizade e carinho, pelas críticas, sugestões e conselhos, pelos bons momentos e pelos puxões de orelha também.

Ao Rodrigo Cabral por toda a ajuda que me deu durante a implementação do sistema.

As minhas duas Veras: Vera Lúcia Sodré Teixeira (Verinha) e Vera Helena Moro Bins Ely, pelo apoio e carinho, principalmente quando a saudade de casa apertou.

Aos amigos Lorival, Paula, Adriana e Albertina, que me acolheram aqui em Florianópolis e foram minha segunda família.

Aos demais amigos do Curso de Pós Graduação em Ciência da Computação, pelo convívio, pelas gargalhadas e pelas lições de vida.

RESUMO

A análise química qualitativa de minerais baseia-se, fundamentalmente, em análises de amostras de características físicas e químicas. Muitas das considerações realizadas pelo químico envolvem a análise de características tais como dureza, brilho, índice de refração, densidade, etc., que são de natureza imprecisa. Um outro aspecto relevante e que merece atenção refere-se ao processamento de informações obtidas através de testes de solubilidade. É difícil determinar os solventes adequados para amostras específicas. Este processo necessita de um conhecimento especializado e envolve ponderações e inferências cognitivas, peculiares ao ser humano. Tal tipo de incerteza é difícil de ser tratada por modelos tradicionais. Para modelar processos de análise química qualitativa de amostras de minerais, um sistema especialista baseado em lógica difusa é proposto.

ABSTRACT

Qualitative chemical analysis is based on, essentiall, phisical and chemical properties of minerals samples. Many judgements performed by the chemist involve the analysis of characteristics such as hardness, luster, refraction index, density, etc., that are surrounded by imprecision. Another important aspect, that deserves attention, is related to the information obtained through the solubility tests. It is very difficult to establish the suitable solvents to specific samples. This process needs an expert knowledge and involves peculiar congnitive ponderations and inferences by the humans being. This kind of uncertainty is very hard to be handled by traditional models. To model the qualitative chemical analysis of minerals samples, an expert system based on fuzzy logic is proposed.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO I

Introdução

1.1. Apresentação

Inteligência Artificial (IA) é a área da Ciência da Computação que, usando uma coleção de técnicas e diferentes linguagens de programação, capacita o computador a se aproximar do raciocínio humano. Os domínios da Inteligência Artificial abrangem a Linguagem Natural, a Robótica, o Aperfeiçoamento das *Interfaces*, a Programação Exploratória e os **Sistemas Especialistas** [Ric91].

Sistemas Especialistas constituem o domínio da Inteligência Artificial de maior sucesso comercial. Usam conceitos de Inteligência Artificial para capacitar os computadores a funcionarem como conselheiros, realizando a tomada de decisão dos especialistas humanos [Mau91].

Sistemas Especialistas emulam o processo de tomada de decisão do ser humano. A lógica tradicional (Booleana) para a maioria das aplicações é insuficiente no que diz respeito a retratação da forma real de pensar do especialista, já que muitos fatos podem assumir valores intermediários entre o Verdadeiro e o Falso. Baseado neste problema, a maioria dos Sistemas Especialistas estão sendo estruturados utilizando a **Lógica Difusa** [Zim92].

A Lógica Difusa é um superconjunto da lógica Booleana a qual foi estendida para tratar com o conceito de "verdade parcial" - valor verdade entre o "completamente verdadeiro" e o "completamente falso" [Zim92].

Um **Sistema Especialista usando Lógica Difusa** é um Sistema Especialista que usa uma coleção de regras difusas para "raciocinar" sobre os dados. O antecedente da regra descreve para que grau a regra se aplica, enquanto a conclusão (consequente da regra) assinala a função de pertinência para cada uma ou mais variáveis de entrada.

A **análise química qualitativa de minerais** baseia-se fundamentalmente em análises de características físicas e químicas. Este processo necessita de um conhecimento especializado e envolve ponderações e inferências cognitivas peculiares ao ser humano. Para modelar este processo, um sistema especialista baseado em lógica difusa foi desenvolvido.

1.2 Definição do Problema

Muitas das considerações realizadas pelo químico envolvem a análise de características tais como dureza, brilho, índice de refração, densidade, etc. Um outro problema relevante, e que merece atenção é o processo de tratamento de informações obtidas através de testes de solubilidade. É difícil determinar os solventes adequados para amostras específicas. Estes tipos de incerteza são difíceis de serem modelados através do uso de modelos tradicionais. Sistemas especialistas que empregam o raciocínio probabilístico podem não se revelar a melhor alternativa, dado o tipo de informação com que se está trabalhando, uma vez que conduzem a uma explosão combinatória.

1.3. Objetivos

O objetivo geral deste trabalho é propor um sistema inteligente, baseado em lógica difusa, para o tratamento de incertezas relacionadas ao processo de análise química qualitativa de amostras de minerais. Como objetivos específicos tem-se:

- Estudar e revisar os conceitos básicos envolvidos.

- Estudar diferentes ferramentas para desenvolvimento de sistemas especialistas.

- Avaliar a *shell* FuzzyCLIPS (*Fuzzy C Language Integrated Production System*) [Nrc94].

- Desenvolver um protótipo de um sistema especialista difuso e aplicá-lo ao processo de análise química qualitativa de amostras de minerais.

1.4. Importância do Trabalho

Trabalhos envolvendo sistemas especialistas têm sido objeto de amplo estudo e interesse [Dep90]; [Lie94]; [Mau91].

Liebowitz [Lie94] destaca a tendência da utilização da teoria dos conjuntos difusos em sistemas especialistas , principalmente em países como Japão. Na Alemanha, as indústrias pesadas estão usando amplamente sistemas especialistas difusos e obtendo excelentes resultados.

A importância deste trabalho baseia-se em:

- permitir a ampliação da utilização de conceitos de Ciência da Computação a outras áreas, mais especificamente, a área de análises químicas.

- permitir a criação de protótipos eficientes e de fácil manipulação pelo usuário final.

- permitir agregar informações próprias do especialista humano em um "sistema inteligente", ampliando, desta forma, o escopo de sua aplicação.

- permitir aumentar a eficiência e a eficácia das análises químicas, sem necessitar tanto do especialista humano.

1.5. Estrutura

Este trabalho está estruturado em sete capítulos, que versam sobre Inteligência Artificial, Sistemas Especialistas Difusos, Lógica Difusa, a ferramenta a ser utilizada, a aplicação desenvolvida e, finalmente, as conclusões e recomendações.

O capítulo 2 apresenta uma visão geral a respeito da Inteligência Artificial e suas subáreas, com um enfoque especial para os Sistemas Especialistas, seus componentes, características básicas, categorias de problemas solucionados e suas diferenças em relação a outros sistemas.

No capítulo 3 discute-se aspectos relativos a Lógica Difusa, nomeadamente, sua evolução, suas diferenças em relação à lógica Booleana, vantagens e utilizações. Discute-se também a teoria dos conjuntos difusos e suas operações.

No capítulo 4, os Sistemas Especialistas Difusos são apresentados através de suas características básicas, a forma como eles modelam a incerteza e suas vantagens em relação a outros sistemas especialistas .

No capítulo 5 estuda-se uma *shell* para desenvolvimento de Sistemas Especialistas: o FuzzyCLIPS . Esta *shell* permite o desenvolvimento de Sistemas Especialistas que utilizam como suporte de inferência a Lógica Difusa.

O capítulo 6 contem uma aplicação prática, envolvendo o processo de análise química qualitativa. São descritos o problema, a solução e os resultados obtidos.

Finalmente, no sétimo capítulo apresentam-se as conclusões e recomendações deste trabalho.

CAPÍTULO II

Inteligência Artificial e Sistemas Especialistas

2.1 - Introdução

A Inteligência Artificial é uma área da Ciência da Computação que tem por objetivo desenvolver programas que resolvam problemas de uma maneira que seja considerada inteligente se realizada por seres humanos [Mau91]. No início dos estudos nesta área, houve um esforço da comunidade científica em criar programas de abrangência geral, para resolver os mais complexos tipos de problemas.

Durante as últimas décadas, o potencial de processamento eletrônico de dados tem sido usado para uma melhoria no grau de decisão do ser humano. Nos finais dos anos 70 e início dos anos 80, sistemas de apoio à decisão encontraram seu caminho dentro do gerenciamento e da engenharia. Os produtos mais recentes são os sistemas especialistas, que têm sido aplicados desde meados dos anos 80 [Zim92].

2.2 Inteligência Artificial

A Inteligência Artificial há muito deixou de ser uma ciência de laboratório em pequena escala e transformou-se em um sucesso tecnológico e industrial. Existe agora um arsenal de técnicas para a criação de programas inteligentes que controlam processos de manufatura, diagnosticam falhas em computadores e doenças em seres humanos, projetam computadores, montam apólices de seguro, jogam xadrez e assim por diante [Ric91].

A Inteligência Artificial abrange basicamente as seguintes subáreas: (i) linguagem natural, onde os pesquisadores trabalham para desenvolver *hardwares* e *softwares* capazes de interagir com a fala humana; (ii) robótica, que desenvolve robôs inteligentes capazes de identificar, mover e manipular objetos; (iii) aperfeiçoamento das *interfaces*: dedica-se a melhoria das *interfaces* existentes, a fim de torná-las mais fáceis de serem usadas; (iv) programação explanatória: emprega técnicas de inteligência artificial para orientar o desenvolvimento de programas, levando o computador (com ajuda humana) a projetar um programa muito mais eficiente em uma dada linguagem de programação usando uma base de conhecimentos dos programas de trabalho atuais [Mau91]; (v) e finalmente, sistemas especialistas, que são programas destinados à resolução de problemas complexos, em domínios especializados, para os quais considera-se necessária a experiência humana [Wat86].

2.2.1 Características das Técnicas Utilizadas em Inteligência Artificial

A inteligência requer conhecimento, porém, este possui algumas propriedades inconvenientes: é volumoso, difícil de ser caracterizado com precisão e está mudando constantemente.

Sendo assim, uma técnica de Inteligência Artificial explora o conhecimento a ser representado, de tal forma que: (i) capture generalizações, porque sem esta característica haverá grande necessidade de memória e atualização; (ii) seja compreendido pelas pessoas que o fornecem; (iii) seja facilmente modificado para corrigir erros e refletir mudanças do mundo e da visão que se tem deste; (iv) seja capaz de ser usado em inúmeras situações, mesmo que não seja totalmente preciso nem esteja totalmente completo; (v) possa ser usado para ajudar a superar seu próprio volume, auxiliando a limitar as várias possibilidades que em geral têm de ser consideradas [Ric91].

2.3 Sistemas Especialistas

Um sistema especialista engloba o conhecimento de um ou mais especialistas humanos em forma de um *software* que pode ser operado em uma variedade de computadores desde o de grande porte até os PC's [Mau91].

A essência de um sistema especialista pode se tornar mais aparente se alguns atributos básicos se fizerem presentes: (i) o sistema tem separado o conhecimento específico do especialista e a metodologia de solução de problemas; (ii) a transferência interativa de conhecimentos pode minimizar o tempo necessário para transferir o conhecimento do especialista para uma base de conhecimento; (iii) a estratégia de controle pode ser simples e transparente ao usuário, isto é, o usuário é capaz de compreender e prever os efeitos de adição, alteração e deleção de itens na base de conhecimento.

2.3.1 Componentes de um Sistema Especialista

Sistemas especialistas possuem basicamente cinco componentes. A conexão entre estes componentes é mostrada na figura 2.1[Dep90].

A máquina de inferência indica a operação a ser realizada sobre o conhecimento contido no sistema especialista. Ela busca as regras necessárias a serem avaliadas, ordena-as de uma maneira lógica e, a partir daí, direciona o processo de inferência. O processo de encaminhar a inferência é feito de acordo com a técnica que foi usada para o armazenamento do conhecimento na base. De uma maneira geral, é um mecanismo de comparação de *strings* com padrões e, se essa comparação for satisfatória, uma outra atitude será tomada, buscando novas regras ou fazendo um outro encaminhamento, até se atingir um determinado objetivo. Há vários processos de inferência, porém, o método mais usado é o da avaliação de regras [Jac90].

A base de conhecimento fornece as características de funcionamento do sistema (encadeamento para frente ou para trás). Armazena os fatos e regras a serem usados.

Subsistema de Aquisição de Conhecimento é usado para alimentar a base de conhecimento. Nele pode-se introduzir, mudar ou deletar regras de acordo com a necessidade. É constituído basicamente por um editor de texto construído especialmente para ser usado pelo especialista e/ou engenheiro do conhecimento na transferência do conhecimento para o sistema [Mau91].

O subsistema de explicação tem por função, elucidar a linha de raciocínio que o sistema especialista usa para chegar a uma conclusão. Ele interage com o usuário esclarecendo-o de como o sistema chegou a determinada conclusão, se isso for solicitado [Mau91].

A *interface* do usuário exibe toda a transação de informações que ocorrem durante a consulta. Pode ser em forma de *menus*, perguntas ou até mesmo ícones. Esta interface também exibe todas as respostas, perguntas e resultados das consultas.

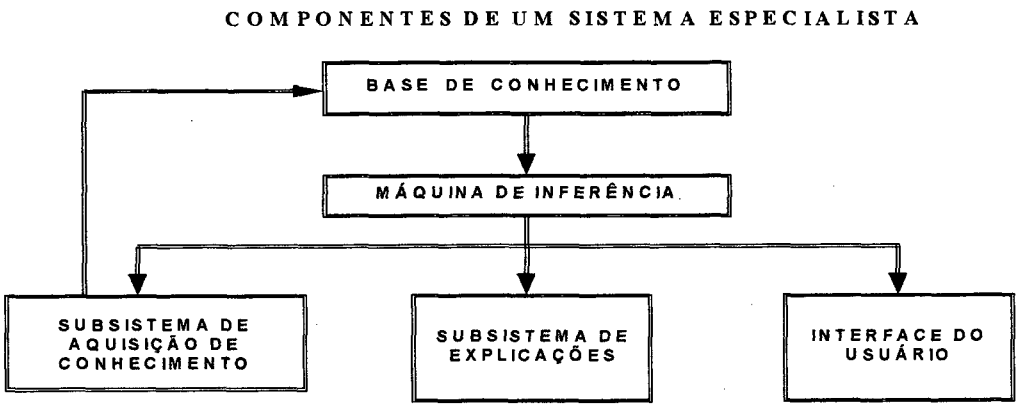

Figura 2.1 - Componentes de um Sistema Especialista.

2.3.2 Características Básicas do Sistema Especialista

A característica básica de um sistema especialista é o alto nível de conhecimento que ele fornece para ajudar na solução de um problema. A flexibilidade do sistema é importante, já que o conhecimento pode ser incrementado de acordo com a necessidade do usuário.

O sistema especialista fornece a solução para um determinado problema e pode explicar em detalhes como uma nova situação pode ser alcançada a partir da anterior. Isto faz com que o usuário avalie o potencial do efeito de novos fatos ou dados e compreenda seu relacionamento para chegar a uma solução final. Similarmente o usuário pode avaliar o efeito de novas estratégias ou procedimentos na solução pelo acréscimo de novas regras ou modificação das regras existentes.

A base de conhecimento que define a eficiência de um sistema especialista pode também fornecer uma característica adicional, uma memória institucional. Se a base de conhecimento foi desenvolvida através de interações com as pessoas chave de um determinado departamento, ela representa a atual política ou procedimento do grupo. Esta compilação de conhecimento se torna um registro permanente da melhor estratégia e métodos usados pela chefia [Dep90].

2.3.3 Engenharia do Conhecimento

Engenharia do conhecimento compreende a aquisição ou eliciação, análise e representação do conhecimento e, hoje, representa o principal gargalo de desenvolvimento dos Sistemas Especialistas.

Existe duas abordagens básicas de engenharia do conhecimento, que são: (i) uma mais psicológica [Har85], que procura obter Sistemas Especialistas que imitam o que o especialista faz. Nesta abordagem o conhecimento modelado, em boa parte, é resultante do processo mental interno do especialista; (ii) uma abordagem baseada em modelos [San90], que parte de modelos de tarefas

básicas, como por exemplo, diagnóstico. Esta abordagem estruturada é mais recente e as ferramentas para a sua implantação estão em fase de desenvolvimento.

O processo de engenharia do conhecimento pode ser dividido em três fases: (i) aquisição ou eliciação do conhecimento; (ii) análise do conhecimento; (iii) representação do conhecimento.

2.3.3.1 Aquisição do Conhecimento

Aquisição do conhecimento é o processo de compreensão da tomada de decisão do especialista humano e transferência deste conhecimento para uma representação formal ou modelo que possa ser observado e validado por outras pessoas [Mau91].

Este processo é considerado o "gargalo" no desenvolvimento de sistemas especialistas. É a etapa mais propensa a erros e é também, a que consome mais tempo. Há no mínimo quatro motivos básicos que a aquisição do conhecimento uma etapa lenta e predisposta a erros [Mau91]:

1. **A tomada de decisão do especialista humano não é compreendida**: quando o especialista não compreende ele mesmo como uma decisão foi tomada, será uma tarefa árdua para o engenheiro do conhecimento tentar modelar o processo de tomada de decisão.
2. **O atual conjunto de técnicas de representação do conhecimento é inadequado para modelar completamente o conhecimento do especialista**: o raciocínio que um computador usa, geralmente, encadeamento para frente ou para trás, na maioria das vezes não coincide com o processo de raciocínio do especialista humano. Assim sendo, o engenheiro do conhecimento tem que criar um modelo que seja adequado à situação.
3. **O processo atual de aquisição do conhecimento a partir do especialista é difícil**: os especialistas podem esquecer como e porque tomaram certas decisões. Eles apenas sabem que elas funcionam. Um bom engenheiro do conhecimento deve ser capaz de ajudar o especialista a compreender o que o processo de tomada de decisão realmente é.

4. **Problemas organizacionais podem tornar o processo muito lento**: se o especialista não está disponível, não há aquisição do conhecimento. Adicionalmente, vários especialistas podem fornecer informações conflitantes.

2.3.3.2 Técnicas para Aquisição do Conhecimento

Algumas das técnicas mais usadas no processo de aquisição do conhecimento são descritas a seguir [Gon86].

- **Entrevista aberta**: deve ser utilizada no início do processo de aquisição do conhecimento, pois ajuda o engenheiro do conhecimento a compreender a natureza, a extensão e a complexidade do conhecimento a ser adquirido. As técnicas de entrevista aberta podem ser divididas em três categorias: (i) técnicas para estabelecer o **rapport** - contato que implica em confiança e segurança; (ii) técnicas para estabelecer o entendimento; (iii) técnicas para expandir o raciocínio.
- ***Brainstorming***: é um método para gerar idéias mais criativas. Em vez de utilizar somente processos lineares e lógicos do lado esquerdo do cérebro, valoriza-se o emprego dos aspectos divergentes, criativos e imaginativos do processo do lado direito do cérebro. O enfoque é se libertar dos pensamentos mais familiares por meio de exploração do impossível e do fantástico.
- **Entrevista dirigida (focada)**: Após se obter uma visão geral do problema, é preciso entrar em detalhes nas áreas de conhecimento mais específicas, relevantes, para o sistema. Exige um planejamento prévio, onde se define os objetivos e os meios para concretizá-los. O entrevistador tem o controle da entrevista e procura buscar detalhes sobre assuntos específicos.
- **Análise de protocolo**: caracteriza-se por uma observação *in loco* do especialista, no momento em que ele está resolvendo o problema em estudo, no seu ambiente de trabalho. Para esta técnica usa-se equipamento de gravação de aúdio e vídeo, com o propósito de evitar que o engenheiro do conhecimento perturbe o processo com sua presença ou interrompa a sequência de raciocínio com suas perguntas.
- **Análise de caso**: o especialista e o engenheiro do conhecimento, em conjunto, analisam um número representativo de casos típicos, reais ou fictícios. A técnica pode ser cansativa, mas produz uma grande quantidade de informações úteis e um conhecimento bastante sólido.

- ***Repertory grid***: permite ao especialista explorar o seu próprio raciocínio. Pode ajudá-lo a iniciar o processo de eliciação do conhecimento, ou retomá-lo quando ocorre um impasse.

2.3.4 Categorias Gerais de Sistemas Especialistas

Os sistemas especialistas são capazes de solucionar vários tipos de problemas. Estes problemas podem ser classificados em diversas áreas [Mau91].

A área de Interpretação abrange os sistemas que trabalham com dedução a partir de descrições de situações relativas aos dados fornecidos pelo usuário. Os sistemas desta área usam sensores de dados e enfrentam grandes dificuldades, já que eles têm de manipular dados que estão com ruídos, estão escassos, incompletos ou errados. Eles necessitam de técnicas especiais para a extração de características de linhas de dados contínuos, linhas de onda, ou figuras, e métodos para representá-los simbolicamente.

A área de Prognósticos abrange os sistemas que trabalham com dedução de consequências plausíveis em uma determinada situação. Estes sistemas algumas vezes usam modelos de simulação para gerar situações ou cenários que podem ocorrer a partir de uma determinada entrada de dados.

A área de Diagnósticos abrange os sistemas que lidam com a dedução das causas do mau funcionamento de um determinado sistema. Eles usam descrições de situações, características comportamentais, ou conhecimento sobre componentes do projeto para inferir as prováveis causas do mau funcionamento do sistema. Estes sistemas podem interagir com o usuário para ajudá-lo a encontrar as falhas e então sugerir ações para corrigi-las.

A área de Projeto abrange os sistemas que lidam com a configuração de novos objetos, a partir de objetos já existentes. Estes sistemas fornecem mecanismos para o desenvolvimento e refinamento de planos até chegar ao projeto desejado. Eles poupam o tempo de busca do melhor plano de ação para a criação da configuração desejada. Esta área está diretamente relacionada à área de Planejamento.

A área de Planejamento abrange os sistemas que lidam com o planejamento de ações. Eles decidem todo o curso de ação antes de agir. Estes sistemas quase sempre rejeitam uma linha de raciocínio particular ou uma porção do plano de ação porque violaram os métodos de construção do planejamento, e voltam ao ponto de partida.

A área de Monitoramento abrange os sistemas que trabalham com a comparação de observações dos resultados das operações de um dado sistema para fornecimento de um resultado final. Eles procuram observar o comportamento que confirme suas expectativas sobre o comportamento normal ou suas hipóteses sobre um possível desvio de comportamento. Estes sistemas lidam diretamente com o tempo, isto pode significar a recordação de todos os valores que um parâmetro adquiriu no sistema em vários intervalos de tempo.

A área de Reparos abrange os sistemas que trabalham com execução de planos para administrar o reparo de problemas.

A área de Depuração abrange os sistemas que lidam com a prescrição de soluções para o mau funcionamento de um dado sistema ou de certas operações. Muitos deles baseiam-se somente em simples tabelas de associação entre os tipos de distúrbios e a solução correspondente, mas o problema é bem mais complexo e requer um sistema de planejamento e avaliação para auxiliá-lo.

A área de Controle abrange sistemas que governam o comportamento geral de outros sistemas (não apenas de computação). São os sistemas mais completos, pois devem interpretar os fatos de uma situação atual, verificando os dados do passado e fazendo uma predição do futuro. Apresentam os diagnósticos de possíveis problemas, formulando um plano ótimo para sua correção. Este plano de correção é executado e monitorado para que o objetivo planejado seja alcançado.

2.4. Diferenças entre os Sistemas Especialistas e os Convencionais

Os sistemas especialistas diferem dos convencionais em vários tópicos. Os sistemas especialistas manipulam o conhecimento, utilizam heurísticas e trabalham com processos de inferência. Os convencionais manipulam dados, utilizam algoritmos e trabalham com processos procedurais.

A linguagem de programação de cada um e a maneira como estas linguagens estão estruturadas representam uma diferença muito importante. As linguagens usadas nos sistemas convencionais são por exemplo: COBOL, FORTRAN, C, PASCAL; que requerem um algoritmo fácil de compreender. A linguagem usada nos sistemas especialistas são as linguagens de programação simbólica como LISP e PROLOG (estas linguagens, porém, podem ser usadas para outros fins). Atualmente também pode ocorrer que após serem escritos em linguagem simbólica, os sistemas especialistas sejam convertidos para linguagens tradicionais como o C. Um contraste entre os dois tipos de linguagem de programação está no fato de que na programação convencional, um algoritmo sempre produz uma solução se são fornecidos dados ao processo. Na programação simbólica, somente será fornecido um resultado se tiverem dados suficientes na base de conhecimento para que o resultado seja alcançado [Mau91].

Há outra importante diferença entre os dois tipos de sistemas. Enquanto os sistemas convencionais são projetados para produzir respostas corretas toda vez que forem solicitados, os

sistemas especialistas, geralmente produzem respostas corretas, mas podem falhar algumas vezes. Neste caso, à primeira vista pode parecer que os sistemas convencionais têm uma certa vantagem em relação aos sistemas especialistas, entretanto, esta vantagem não existe. Os sistemas convencionais também cometem erros na realização de tarefas complexas como as realizadas pelos sistemas especialistas. Os sistemas especialistas, como os humanos, cometem equívocos, mas ao contrário dos sistemas convencionais, eles têm a capacidade de aprender com os próprios erros [Wat86].

2.5 Desenvolvimento de Sistemas Especialistas

Os sistemas especialistas têm sido cada vez mais pesquisados e seu desenvolvimento aumenta a cada dia em vários países do mundo, nas mais diversas áreas de pesquisa [Lie94]. De acordo com Zurada [Zur94], o número de patentes requeridas por sistemas especialistas esta em franco crescimento.

Um breve levantamento a respeito do desenvolvimento de sistemas especialistas em países como Estados Unidos, Japão, Alemanha, Espanha e Nova Zelândia [Lie94], é descrito a seguir.

Cada vez mais, o mercado de sistemas especialistas nos Estados Unidos ganha espaço. As tendências mais comuns são: (i) um movimento contínuo em direção à integração e aos sistemas híbridos; (ii) ênfase para o problema de "solução de negócios"; (iii) crescimento da tendência de criação de sistemas de informação "ativos", amplas bases de conhecimento, compartilhamento deste conhecimento e sistemas inteligentes híbridos; (iv) necessidade de fornecer suporte de alto nível para pesquisa em Inteligência Artificial; (v) uso de metodologias estruturadas para desenvolvimento de sistemas especialistas.

No Japão, os fabricantes de computador e as principais companhias do país têm investido no desenvolvimento de sistemas especialistas, sendo que a cada ano, aumenta o número de empresas de médio e pequeno porte especializadas em sistemas deste tipo. Os sistemas especialistas para auxílio em diagnósticos, planejamento e escalonamento, além dos direcionados à indústria pesada (aço, eletromecânica, automobilística, óleo, papel, construção, etc.) são cada vez mais populares. As aplicações utilizando lógica difusa estão conquistando uma grande fatia do mercado, principalmente no que diz respeito aos produtos eletrodomésticos.

Os sistemas especialistas têm sido bastante utilizados nas indústrias pesadas da Alemanha. Além disso, as aplicações baseadas em lógica difusa estão cada vez mais populares.

A Espanha demonstra uma grande preocupação no sentido de transferir a tecnologia de sistemas especialistas dos laboratórios das universidades para a indústria. A Universidade Politécnica tem oferecido um programa de mestrado em Engenharia do Conhecimento, uma vez que a aquisição do conhecimento envolve todas as etapas de construção de um sistema especialista.

Há na Nova Zelândia uma forte tendência em direção ao uso da tecnologia de sistemas especialistas em processos relacionados à automação e reengenharia. A Universidade de Utrecht, na Holanda, vem oferecendo no Departamento de Ciências da Computação um programa de mestrado voltado exclusivamente para esta área.

CAPÍTULO III

Lógica Difusa

3.1 Introdução

A lógica difusa é uma poderosa técnica para a solução de problemas, com uma vasta aplicabilidade, especialmente nas áreas de controle e tomada de decisão. Em geral, ela é muito usada em problemas que não são definidos facilmente por modelos matemáticos práticos.

A força da lógica difusa deriva da sua habilidade em criar conclusões e gerar respostas baseadas em informações vagas, ambíguas e qualitativamente incompletas ou imprecisas. Neste aspecto, sistemas de base difusa têm a habilidade de raciocinar de forma semelhante à dos humanos. Seu comportamento é representado de uma maneira muito simples e natural, levando à construção de sistemas compreensíveis e de fácil manutenção. Além disso, o uso da lógica difusa requer muito menos memória e capacidade do computador, do que os métodos convencionais, permitindo assim, sistemas menores e mais poderosos.

A Teoria dos Conjuntos Difusos é uma ampliação da teoria tradicional para resolver os paradoxos gerados a partir da classificação "tudo ou nada" da lógica clássica. Tradicionalmente, uma proposição lógica tem dois extremos: ou "completamente verdadeiro" ou "completamente falso". Entretanto, no mundo difuso, uma premissa varia em graus de verdade de 0 a 1, o que leva a ser parcialmente verdadeira ou parcialmente falsa [Bez93].

Pela incorporação deste conceito de "graus de verdade", a lógica difusa estende a lógica clássica em dois caminhos. Primeiro, os grupos são rotulados qualitativamente (usando termos lingüístico, tais como : alto, morno, ativo, perto, etc.) e os elementos destes grupos são

caracterizados variando o grau de pertinência. Por exemplo, um homem de 1.80m e um homem de 1.75m são membros do grupo "ALTO", embora o homem de 1.80m tenha um grau de pertinência maior neste grupo.

As entradas, cálculos e saídas da lógica difusa envolvem números precisos, manipulados de uma maneira fundamentada em inferência lógica. O que é vago na lógica difusa é a expressão lingüísticas de um problema e sua solução, não a representação numérica. Mas a expressão lingüísticas vaga ou geral de um problema pode ser muito poderosa. Ela leva o usuário a desenvolver ou mesmo compreender um modelo numérico detalhado. Sendo assim, a lógica difusa pode ser aplicada em problemas de complexidade lógica e combinatória, para os quais é impossível construir modelos numéricos por causa do enorme número de combinações possíveis. O poder descritivo da lógica difusa vem da sua habilidade de deixar os especialistas expressarem seus conhecimentos em uma linguagem que eles compreendam.

3.2 Teoria dos Conjuntos Difusos

A teoria dos conjuntos difusos procura generalizar a noção clássica de conjuntos e proposições para tratar a incerteza. Ela fornece modelos matemáticos de trabalho nos quais o conceito de "vago" pode ser estudado de maneira precisa e rigorosa [Zim92].

3.2.1 As Propriedades dos Conjuntos Difusos

Visto que os conjuntos difusos são generalizações dos conjuntos tradicionais (*crisp*), eles possuem propriedades semelhantes [Lie94].

Dados os três conjuntos difusos: $\underline{A}$, $\underline{B}$ e $\underline{C}$, e sendo X o universo, pode-se definir para esses conjuntos as propriedades contidas na Tabela 1.

Tabela 1 - Propriedades dos Conjuntos Difusos.

Comutatividade	$\underline{A} \cup \underline{B} = \underline{B} \cup \underline{A} \quad \underline{A} \cap \underline{B} = \underline{B} \cap \underline{A}$
Associatividade	$\underline{A} \cup (\underline{B} \cup \underline{C}) = (\underline{A} \cup \underline{B}) \cup \underline{C} \quad \underline{A} \cap (\underline{B} \cap \underline{C}) = (\underline{A} \cap \underline{B}) \cap \underline{C}$
Distributividade	$\underline{A} \cup (\underline{B} \cap \underline{C}) = (\underline{A} \cup \underline{B}) \cap (\underline{A} \cup \underline{C}) \quad \underline{A} \cap (\underline{B} \cap \underline{C}) = (\underline{A} \cap \underline{B}) \cup (\underline{A} \cap \underline{C})$
Identidade	$\underline{A} \cap X = \underline{A} \quad \underline{A} \cap 0 = 0 \quad \underline{A} \cup X = X$
Idempotência	$\underline{A} \cup \underline{A} = \underline{A} \quad \underline{A} \cap \underline{A} = \underline{A}$
Transitividade	Se $\underline{A} \subseteq \underline{B} \subseteq \underline{C}$, então $\underline{A} \subseteq \underline{C}$

3.2.2 Operações com Conjuntos Difusos

As operações com conjuntos difusos são definidas através de sua função de pertinência (μ). A seguir, as operações básicas são definidas nas tabelas 2 e 3.

Tabela 2 - Operadores para Intersecção entre Conjuntos Difusos.

Intersecção: para $\underline{C} = \underline{A} \cap \underline{B}$	
Intersecção de Zadeh	$\mu_{\underline{C}}(x) = min\{\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)\}, x \in X$
Diferença Limitada	$\mu_{\underline{C}}(x) = max\{0, \mu_{\underline{A}}(x) + \mu_{\underline{B}}(x) - 1\}$
Produto Algébrico	$\mu_{\underline{C}}(x) = \mu_{\underline{A}}(x) * \mu_{\underline{B}}(x)$
Produto de Einstein	$\mu_{\underline{C}}(x) = \frac{\mu_{\underline{A}} * \mu_{\underline{B}}}{2 - [\mu_{\underline{A}} + \mu_{\underline{B}} - \mu_{\underline{A}} * \mu_{\underline{B}}]}$
Produto de Hamacher	$\mu_{\underline{C}} = \frac{\mu_{\underline{A}} * \mu_{\underline{B}}}{\mu_{\underline{A}} + \mu_{\underline{B}} - \mu_{\underline{A}} * \mu_{\underline{B}}}$

Tabela 3 - Operadores para União entre Conjuntos Difusos.

União: para $C = A \cup B$	
União de Zadeh	$\mu_C(x) = max\{\mu_A(x), \mu_B(x)\}, x \in X$
Soma Limitada	$\mu_C(x) = min\{1, \mu_A(x) + \mu_B(x)\}$
Soma Algébrica	$\mu_C(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) * \mu_B(x)$
Soma de Einstein	$\mu_C(x) = \frac{\mu_A + \mu_B}{1 + \mu_A * \mu_B}$
Soma de Hamacher	$\mu_C = \frac{\mu_A + \mu_B - 2\mu_A * \mu_B}{1 - \mu_A * \mu_B}$

O **Complementar** de um conjunto difuso A é definido por : $\mu_{\neg A} = 1 - \mu_A$

3.2.3 Relações entre Conjuntos Difusos

A relação entre conjuntos representa a presença ou ausência de associação, interação ou interconectividade entre os elementos de dois ou mais conjuntos. Este conceito pode ser generalizado a fim de alcançar vários graus de relação entre os elementos. Estes graus de associação podem ser representados pelo grau de pertinência em uma relação difusa da mesma maneira que o grau de pertinência é representado em um conjunto. Por exemplo: Sendo os conjuntos X e Y $\subseteq$ R, então , $R = \{((x,y)\, \mu_R(x,y))|\ (x,y) \subseteq X \times Y\}$ é chamada uma relação difusa em X x Y. Então, uma relação difusa é um conjunto difuso definido em um Produto Cartesiano de conjuntos *crisp* (X e Y por exemplo), onde cada tupla (x,y) tem apenas um grau de pertinência dentro de uma relação. Este grau de pertinência indica a intensidade da relação entre os elementos da tupla [Tur91].

Uma relação difusa pode também ser representada por uma matriz n-dimensional de pertinência cuja entrada corresponde a n-tuplas no conjunto universo. Estas entradas recebem os valores representando os graus de pertinência das n-tuplas correspondentes.

3.2.3.1 Composições das Relações Difusas

As relações difusas em diferentes espaços podem ser combinadas umas com as outras através das operações de **"composição"**. As operações mais utilizadas e que fornecem resultados mais próximos do esperado (dependendo da aplicação - em campos cognitivos isto pode não ocorrer) são as operações de *max-min, max-product, max-average*, como mostra a tabela 4.

Sendo $\underline{R}_1(x,y)$, $(x,y) \in XxY$ e $\underline{R}_2(y,z)$, $(y,z) \in YxZ$ duas relações difusas, tem-se três composições básicas, como mostrado na tabela 4:

Tabela 4 - Composições das Relações Difusas.

max-min	$\underline{R}_1 \circ \underline{R}_2 = \{[(x,y), \max_y \{\min\{\mu_{\underline{R}_1}(x,y), \mu_{\underline{R}_2}(y,z)\}] \mid x \in X, y \in Y, z \in Z\}$
max-product	$\underline{R}_1 \overset{\circ}{*} \underline{R}_2 = \{[(x,z), \max_y(\mu_{\underline{R}_1}(x,y) * \mu_{\underline{R}_2}(y,z))] \mid x \in X, y \in Y, z \in Z\}$
max-average	$\underline{R}_1 \overset{\circ}{av} \underline{R}_2 = \{[(x,z), 1/2 \max_y(\mu_{\underline{R}_1}(x,y) + \mu_{\underline{R}_2}(y,z))] \mid x \in X, y \in Y, z \in Z\}$

3.3 Raciocínio Aproximado Utilizando a Lógica Difusa

A lógica difusa é um superconjunto da lógica convencional (Booleana), que foi estendido para manipular o conceito de verdade parcial - valores verdade entre "completamente verdadeiro" e "completamente falso".

Na teoria clássica dos conjuntos, um subconjunto U de um conjunto S pode ser definido como um mapeamento de elementos de S para os elementos do conjunto {0,1}, $U : S \rightarrow \{0,1\}$. Este mapeamento pode ser representado como um conjunto de pares ordenados, com exatamente um par ordenado presente para cada elemento de S. O primeiro elemento do par ordenado é um elemento do conjunto S, e o segundo é um elemento do conjunto {0,1}. O valor zero é usado para representar uma "não pertinência" completa; o valor um é usado para representar pertinência completa; e valores entre eles são usados para representar graus intermediários de pertinência. A veracidade ou falsidade da declaração "x está em U" é determinada encontrando o par ordenado cujo primeiro elemento é x. A declaração é verdadeira se o segundo elemento do par ordenado é 1, e a declaração é falsa se ele é zero [Lie94].

Um subconjunto difuso F de um conjunto S pode ser definido como sendo um conjunto de pares ordenados, cada um com o primeiro elemento de S, e o segundo elemento, do intervalo [0,1], sendo exatamente um par ordenado presente para cada elemento de S. Isto define um mapeamento entre os elementos do conjunto S e valores no intervalo [0,1]. O conjunto S é referido como o universo de discurso para o subconjunto difuso F. Freqüentemente, o mapeamento é descrito como uma função de pertinência de F. O grau para o qual a declaração "x está em F" é verdadeira, é determinado encontrando o par ordenado cujo primeiro elemento é x. O grau de verdade da declaração é o segundo elemento do par ordenado [Bez93]. A figura 3.1 mostra as diferenças entre as duas lógicas.

DIFERENÇAS ENTRE AS LÓGICAS DIFUSA E BOOLEANA

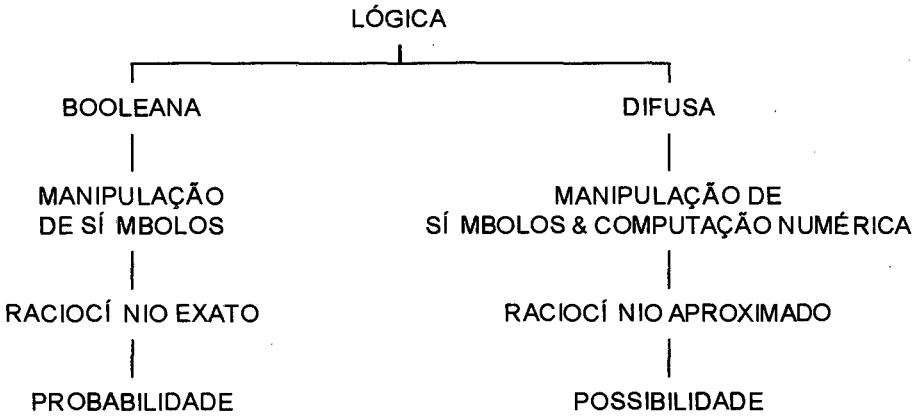

Figura 3.1 - Diferenças entre a Lógica Difusa e a Lógica Booleana.

3.4 Função de Pertinência

Nos conjuntos tradicionais, a função que mapeia a pertinência ou não de um elemento a um conjunto (f(x)) - função característica, assume o valor 0 se x não pertence a um determinado conjunto, e 1, se x pertence a esse conjunto. Quando se trata de um conjunto difuso, f(x) chama-se função de pertinência e não tem de ser necessariamente limitada pelos valores entre 0 e 1. Se um conjunto difuso é uma generalização de um conjunto clássico, a função de pertinência é uma generalização da função característica. As funções de pertinência usadas na maioria das aplicações tendem a ser triangulares ou trapezoidais (figura 3.11).

Segundo Bastos [Bas94], para construção de funções de pertinência deve-se verificar as seguintes propriedades:

- todas as funções de pertinência são contínuas;
- todas as funções de pertinência mapeam um intervalo [a,b] $\rightarrow$ [0,1] $\rightarrow$ μ[a,b] $\rightarrow$[0,1];
- as funções de pertinência são:

- monotonicamente crescentes, monotonicamente decrescente, ou subdividida em parte crescente e parte decrescente;

- as funções de pertinência monótonas sobre um intervalo completo são:
  - funções convexas, ou
  - funções côncavas, ou
  - existe um ponto "c" no [a,b] tal que [a,c] é convexo e [c,b] é côncava.
- funções monotonicamente crescentes tem a propriedade $\mu(a) = 0$ e $\mu(b) = 1$, enquanto funções monotonicamente decrescentes tem a propriedade $\mu(a) = 1$ e $\mu(b) = 0$;
- as funções de pertinência devem apresentar uma forma linear ou devem ser linearizadas.

As funções de pertinência possuem características especiais, tais como: centro da função, suporte, fronteiras, normalização e convexidade. A figura 3.2 auxilia nesta descrição.

O **centro** de uma função de pertinência para um conjunto difuso $\underline{A}$ qualquer é definido como a região do universo que é caracterizada pela pertinência total a este conjunto. Isto é, o centro é formado por todos os elementos do universo que possuem $\mu_{\underline{A}}(x) = 1$ [Cox94].

O **suporte** de uma função de pertinência para um conjunto difuso $\underline{A}$ qualquer, é definido como a região do universo que é caracterizada pela pertinência diferente de zero neste conjunto. Isto é, o suporte é formado pelos elementos do universo que possuem $\mu_{\underline{A}}(x) \neq 0$ [Cox94].

As **fronteiras** (ou limites) de uma função de pertinência para um conjunto difuso $\underline{A}$ qualquer são definidas como as regiões do universo que contêm os elementos detentores de pertinência não nula, mas também, não completa. Isto é, as fronteiras são compreendidas pelos elementos do universo onde $0 < \mu_{\underline{A}}(x) < 1$. Estes elementos do universo são os que possuem uma pertinência parcial no conjunto difuso $\underline{A}$.

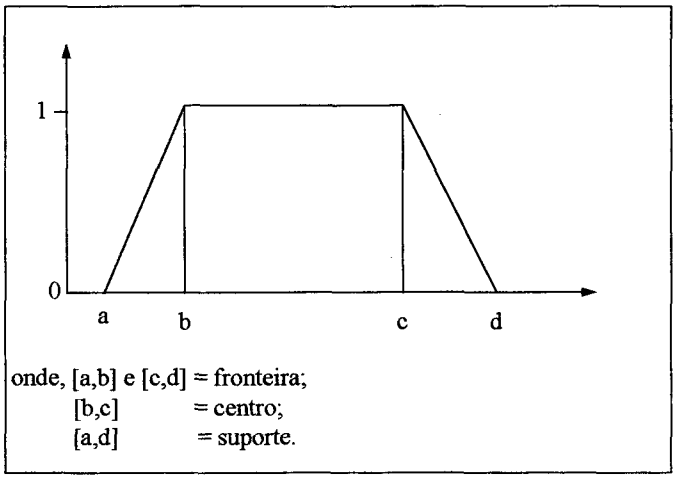

Figura 3.2 - Características Básicas de uma Função de Pertinência.

Conjunto difuso normalizado é aquele cuja função de pertinência tem no mínimo um elemento no universo cujo valor de pertinência é igual a 1. Para os conjuntos difusos onde um e somente um elemento tem uma pertinência igual a 1, este elemento é tipicamente chamado "protótipo" do conjunto. **Conjunto difuso não normalizado** pode ser normalizado dividindo o grau de pertinência de cada elemento, pelo maior valor de pertinência encontrado no conjunto. A figura 3.3 ilustra conjuntos difusos normalizados e não normalizados.

Conjunto difuso convexo é caracterizado por uma função de pertinência onde os valores de pertinência são estritamente monotônicos e crescentes, ou cujos valores de pertinência são estritamente monotônicos e decrescentes. Em outras palavras, se para todos os elementos pertencentes a um conjunto difuso $\underline{A}$ contínuo $x < y < z$ e $\mu_{\underline{A}}(y) \geq min[\mu_{\underline{A}}(x), \mu_{\underline{A}}(z)]$, então $\underline{A}$ é dito como sendo um conjunto difuso convexo. A figura 3.4 mostra um conjunto difuso convexo e um conjunto difuso não convexo.

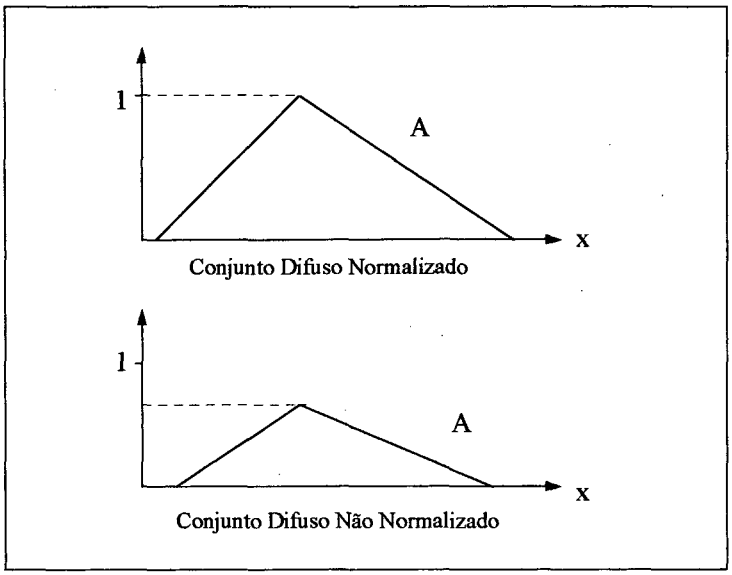

Figura 3.3 - Conjuntos Difusos Normalizados e Não Normalizados.

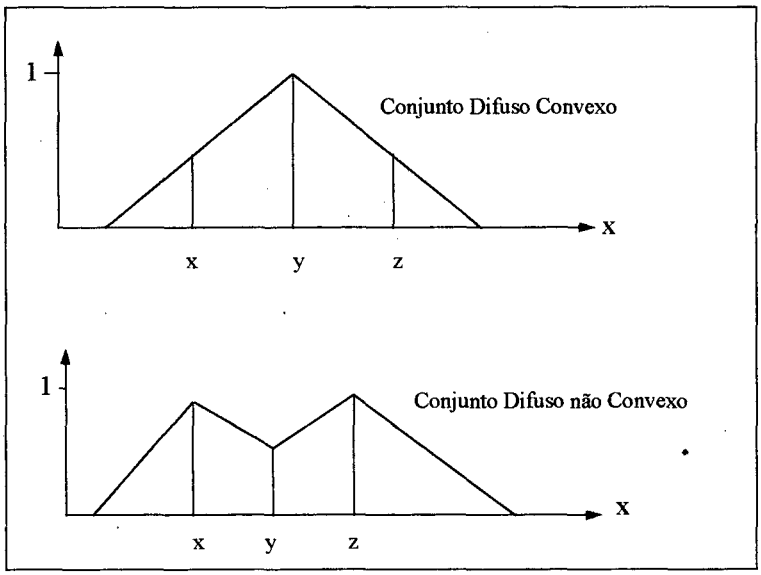

Figura 3.4 - Conjuntos Difusos Convexos e Não Convexos.

Quanto à convexidade, pode-se verificar uma propriedade especial: sendo dois conjuntos difusos $\underline{A}$ e $\underline{B}$, convexos, a intersecção destes dois conjuntos difusos é também um conjunto difuso

convexo. Isto é, para $\underline{A}$ "and" $\underline{B}$, onde ambos são convexos, então $\underline{A} \cap \underline{B}$ é também convexo, como mostra a figura 3.5.

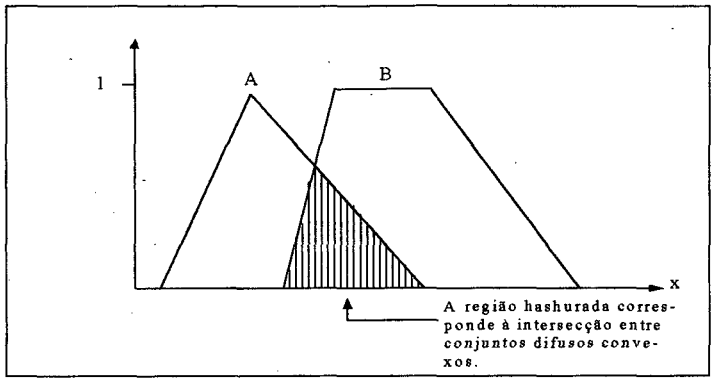

Figura 3.5 - Intersecção entre Conjuntos Difusos Convexos.

3.4.1 Determinação dos Valores de Pertinência

Os métodos para a determinação dos valores de pertinência são bastante variados e podem ser enquadrados em uma das seguintes categorias: método de eliciação e avaliação subjetivas; método de conversão de freqüências ou probabilidades; método de medidas físicas.

No método de eliciação e avaliação subjetivas, os conjuntos difusos são usualmente utilizados com o propósito de modelar os estados cognitivos das pessoas; eles podem ser determinados a partir de procedimentos de eliciação simples ou sofisticados; desde assuntos simples até conhecimentos específicos de uma determinada área.

Muitas aplicações da lógica difusa usam medidas físicas, mas nunca medem diretamente o grau de pertinência. Ao invés disso, uma função de pertinência é fornecida por outro método, e então, os graus de pertinência individual dos dados são calculados a partir dela.

3.5 Representação de Conjuntos e Números Difusos

A superfície de um conjunto difuso, ou seja, a parte do conjunto que define a função de pertinência, pode assumir qualquer formato. Geralmente a superfície é uma linha contínua unindo a margem esquerda à margem direita. Os contornos de um conjunto difuso representam as propriedades semânticas do conceito em estudo, então, quanto mais próxima a superfície estiver do comportamento de um fenômeno físico ou conceitual, melhor será a representação do mundo real [Cox94].

A representação de um número difuso pode ser feita de várias maneiras. A figura 3.6 mostra o conceito de "Próximo a 5" modelado por uma curva "Sino". Já a figura 3.7 mostra o mesmo conceito, modelado como um espaço triangular. Quando sobrepõe-se estas duas curvas, a diferença é aparente, porém, os modelos difusos são praticamente insensíveis a este tipo de fato. Tal característica torna os modelos difusos bastante robustos.

A maneira mais simples de representação de um conjunto difuso é a superfície linear. É sempre usada quando o conceito é desconhecido ou muito pouco compreendido e não constitui um número difuso. Há dois estados para um conjunto difuso linear. O conjunto crescente e o decrescente. O crescente começa no valor do domínio de menor pertinência e se desloca para a direita em direção ao de maior pertinência. O decrescente é exatamente o contrário, como mostra a figura 3.8.

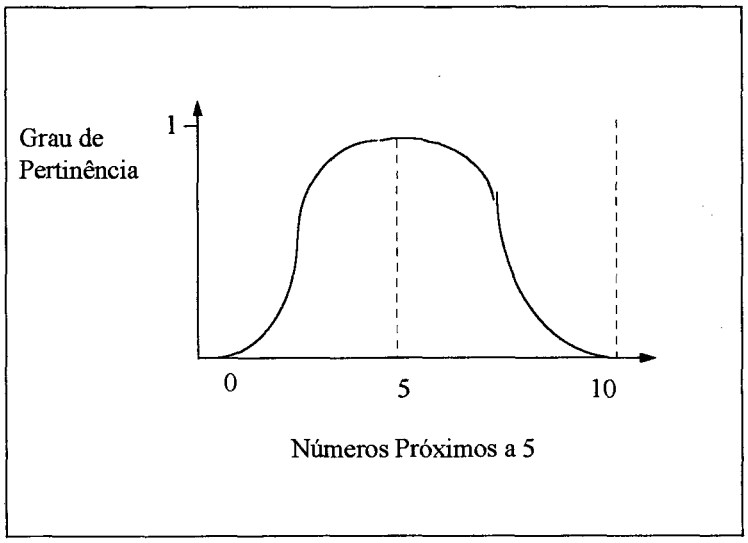

Figura 3.6 - Representação de "Números Próximos a 5", Usando uma Curva "Sino".

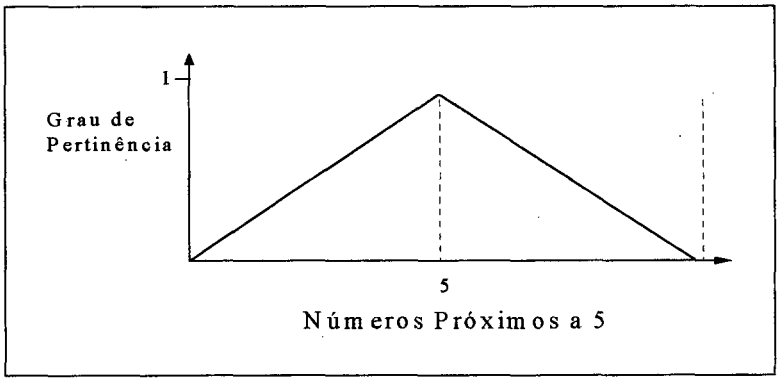

Figura 3.7 - Representação de "Números Próximos a 5", Usando Espaço Triangular.

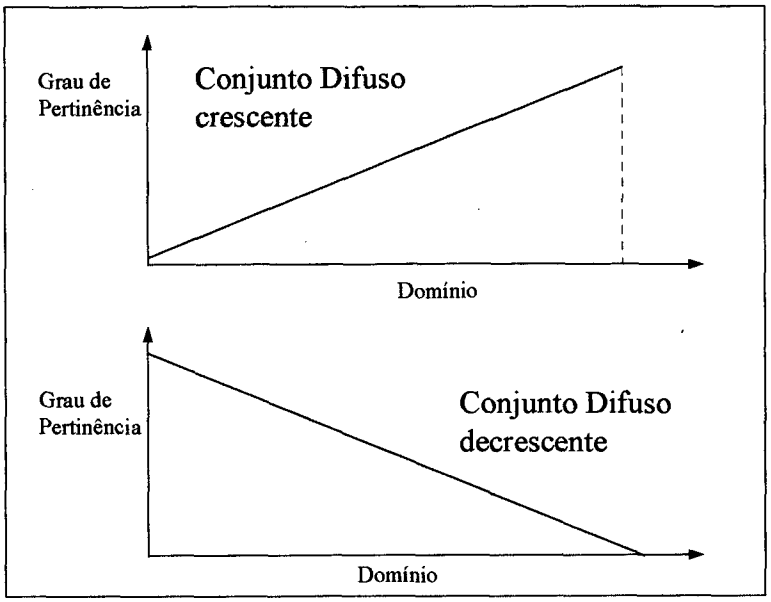

Figura 3.8 - Conjuntos Difusos Lineares Crescente e Decrescente.

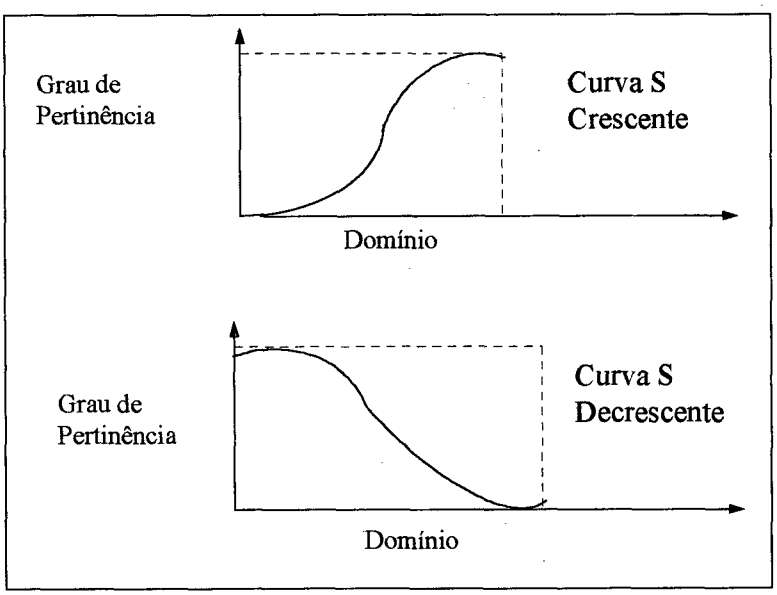

Figura 3.9 - Representação da Curva "S" na Forma Crescente e na Forma Decrescente.

As curvas em forma de sino representam os chamados "números difusos". Há três importantes classes de curvas em forma de sino (figura 3.6): PI, Beta e Gauss. A diferença entre as três curvas está na inclinação da curva e nos valores dos pontos externos.

Um número difuso representa uma aproximação de algum valor e geralmente fornece um conjunto de trabalho melhor do que o correspondente em valores *crisp*. Além disso, os números difusos desempenham um importante papel na definição dos modificadores (pouco, muito, etc.).

A curva PI é a forma preferida de representação dos números difusos. Já a curva Beta tem o formato mais fechado que a PI. A incapacidade de prever exatamente o formato da curva em forma de sino resultante, torna esta curva difícil de ser usada [Cox94].

Algumas vezes as funções ou curvas padrão falham na captura da semântica de um modelo particular ou a representação usando os padrões torna-se ineficiente. Isto ocorre especialmente quando um conjunto difuso é usado para representar independentemente o estado de uma variável. Neste caso, tem-se que definir um formato arbitrário para o conjunto difuso, como mostrado na figura 3.10.

Um conjunto difuso pode ser especificado por uma simples lista dos valores de pertinência no intervalo [0,1]. O valor absolutamente verdadeiro é colocado no ponto médio do domínio. O exemplo típico disto é a função triangular. Quando há mais que um valor absolutamente verdadeiro, a função triangular é expandida, tomando a forma de uma função trapezoidal, como mostra a figura 3.11.

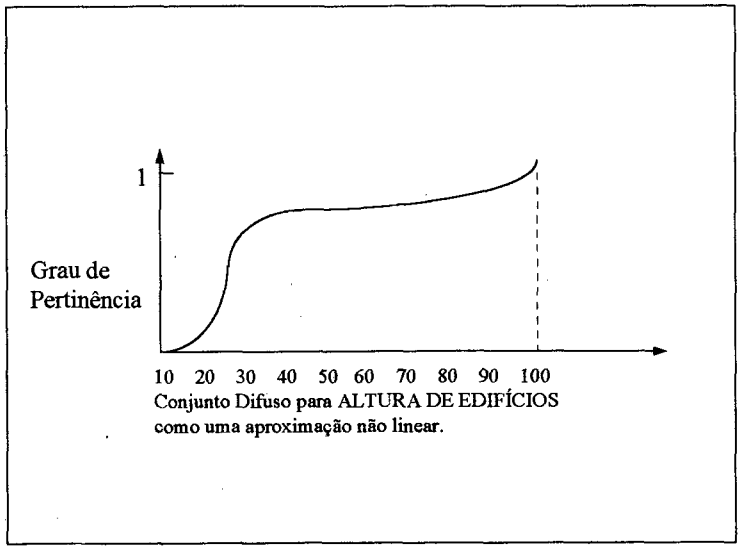

Figura 3.10 - Exemplos de Conjuntos Difusos que Não Usam Funções Padrão.

3.6 Fuzificação

Fuzificação é um processo que consiste em assinalar e calcular um valor para representar o grau de pertinência da entrada em um ou mais "conjuntos difusos". Cada valor de entrada tem um grau de pertinência em cada um dos grupos. O grau de pertinência é um ponto da função de pertinência que é definida baseada na experiência ou na intuição [Bez93].

As funções de pertinência mudam várias vezes à medida que o sistema é levado a aprimorar as respostas em relação às variáveis de entrada. Geralmente, uma vez que o sistema está em operação, as funções de pertinência não mudam. Algumas funções trapezoidais e triangulares são usadas para definir a pertinência em conjuntos difusos, mas deve-se levar em conta o problema em questão. O tipo e a quantidade de funções de pertinência usados em um sistema dependem de alguns fatores tais como: precisão, estabilidade, facilidade de implementação, manipulação e manutenção, dentre outros.

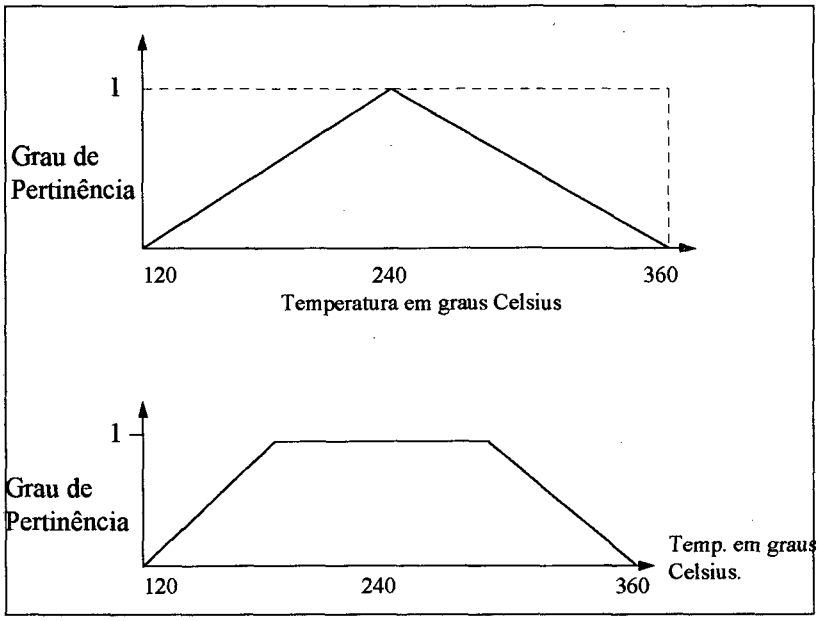

Figura 3.11 - Conjuntos Difusos Triangular e Trapezoidal.

As funções de pertinência triangulares e trapezoidais são muito comuns e têm provado serem bem eficientes. As fronteiras dos conjuntos difusos devem situar-se no eixo X, cobrindo toda a faixa, ou universo em estudo, para o sistema de entrada. Mapeando o eixo Y, está a faixa de 0 a 1, que representa o grau de pertinência de um valor de entrada em relação a um determinado conjunto difuso.

A sobreposição das fronteiras dos conjuntos difusos é desejável e é a chave para as operações do sistema. Este fato permite pertinências em múltiplos - e aparentemente contraditórios - conjuntos.

3.7 Avaliação das Regras

Para governar o comportamento do sistema, o projetista desenvolve um conjunto de regras que têm a forma IF - THEN. O lado IF de uma regra contem uma ou mais condições, chamadas "antecedentes", o lado THEN contem uma ou mais ações chamadas conseqüentes. Os antecedentes

das regras correspondem diretamente aos graus de pertinência calculados durante o processo de fuzificação [Tur93].

Por exemplo, considerando uma regra de um sistema de bolsa de valores mostrado na figura 3.12.

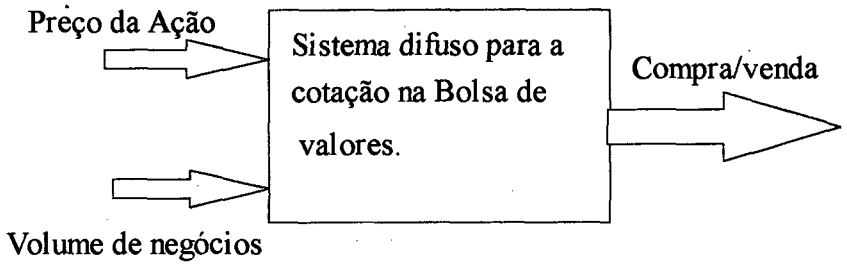

Figura 3.12 - Exemplo dos Antecedentes e Conseqüentes de Uma Regra no Sistema de Bolsa de Valores.

Este sistema apresenta os conjuntos difusos representados na figura 3.13.

O sistema utiliza regras de produção (regras que são organizadas na forma IF-THEN), como por exemplo: IF **preço da ação decrescente** AND **volume de negócios muito pesado** THEN a **ordem é vender**. As duas condições "preço da ação" é decrescente, e "volume de negócios" é pesado são antecedentes da regra. Cada antecedente tem um grau de pertinência indicado para ele como resultado da fuzificação. A ação da regra (ou saída difusa) é para "**vender**" ações. Durante a avaliação das regras, a intensidade é calculada com base em valores dos antecedentes e estão indicadas para saídas difusas da regra.

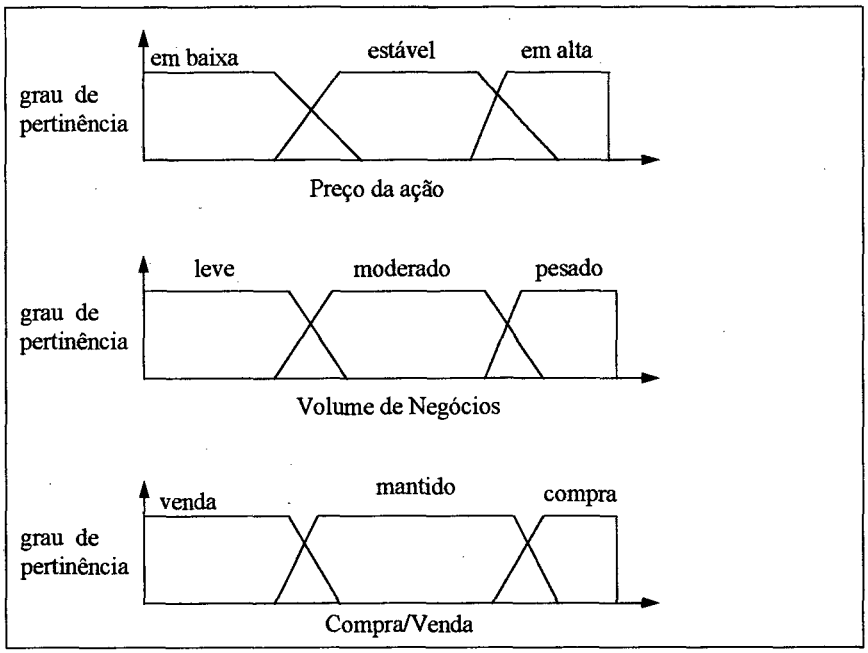

Figura 3.13 - Conjuntos Difusos Utilizados no Sistema da Bolsa de Valores.

3.8 Defuzificação

Defuzificação é um processo utilizado para converter o conjunto difuso de saída de um sistema em um valor *crisp* correspondente. Ele é necessário por duas razões: decifrar o significado das ações difusas, tal como a ordem é "vender", usando funções de pertinência; e para resolver conflitos entre ações adversas tais como: a ordem é vender e a ordem é segurar, as quais podem ter sido selecionadas por certas condições durante a avaliação das regras.

Há muitos métodos utilizados para defuzificação. Os dois métodos mais comuns são os métodos do **centróide** e da **média dos máximos** [Cox94]. O centróide, também chamado de centro de gravidade (COG ou Momento), tem a vantagem de produzir regularmente uma variação na saída do controlador, mas é muitas vezes criticado como dando um peso insuficiente para o conseqüente das regras que concordam e é obrigado a reforçar cada uma. Já a média dos máximos (MOM) concentra-se nos valores onde a distribuição de possibilidade chega ao máximo. Seu algoritmo é criticado por produzir poucas saídas regulares, mas tem como vantagem a grande velocidade para cálculos em ponto flutuante.

Por exemplo, assumindo que as variáveis x, y, e z têm valores no intervalo de [0,10], e que as seguintes funções de pertinência e regras são definidas:

```
baixo(t) = 1 -(t/10)
alto(t) = t/10
regra 1: se x é baixo e y é baixo, então z é alto
regra 2: se x é baixo e y é alto, então z é baixo
regra 3: se x é alto e y é baixo, então z é baixo
regra 4: se x é alto e y é alto, então z é alto
```

A figura 3.14 ilustra os conjuntos difusos relativos a baixo e alto.

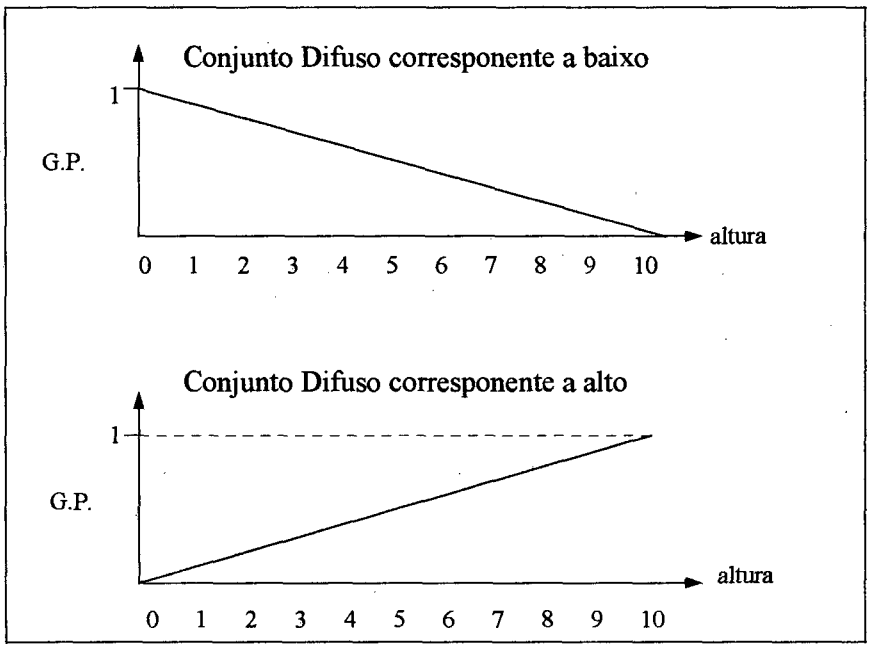

Figura 3.14 Conjuntos Difusos Relativos a Baixo e Alto.

Ao invés de determinar um simples valor de variáveis de saída z, cada regra determina um subconjunto difuso inteiro (baixo ou alto).

Neste exemplo, baixo(t) + alto(t) = 1.0 para todos os t. O valor de t para o qual baixo(t) é máximo é o mesmo valor de t para o qual alto(t) é mínimo, e vice-versa. As mesmas funções de pertinência são usadas para todas as variáveis.

No subprocesso de fuzificação, as funções de pertinência definidas nas variáveis de entrada são aplicadas nos seus valores atuais, para determinar o grau de verdade para cada premissa da regra. O grau de verdade para as premissas das regras algumas vezes refere-se ao corte alfa. Se a premissa da regra tem um grau de verdade diferente de zero (se a regra se aplica a todos...) então a regra DISPARA [Tur93].

No subprocesso de inferência, o valor verdade para a premissa de cada regra é calculado, e aplicado para a parte de conclusão de cada regra. Isto resulta em um subconjunto difuso a ser determinado para cada variável de saída para cada regra.

MIN e PRODUTO são dois métodos de inferência ou regras de inferência aplicados à intersecção. Na inferência MIN, a função de pertinência de saída é retirada em um peso correspondente ao grau de verdade calculado para a premissa da regra. Isto corresponde à interpretação tradicional do AND lógico difuso. Na inferência do PRODUTO, a função de pertinência é determinada pelo cálculo do grau de pertinência estabelecido para a premissa da regra [Kos92].

No caso dos valores x = 0.0 e y = 3.2 para a regra 1, o grau de verdade da premissa é maior que 0.68. Para esta regra, a inferência MIN irá assinalar o subconjunto difuso Z (figura 3.15) definido pela função de pertinência:

regra1(z) = { z/10, se z <= 6.8 $\quad 6.8/10 = 0,68$
0.68, se z >= 6.8}

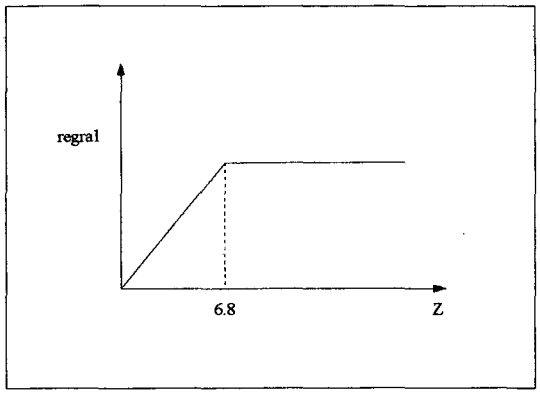

Figura 3.15 Subconjunto Difuso criado pela Inferência MIN

Para as mesmas condições, a inferência do PRODUTO irá assinalar z para o subconjunto difuso (figura 3.16) definido pela função de pertinência:

regra1(z) = 0.68 * alto(z) = 0.068 * z $0{,}68/10 = 0{,}068$

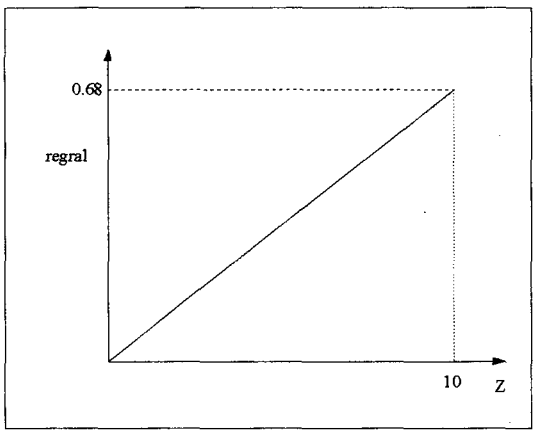

Figura 3.16 Subconjunto Difuso criado pela Inferência PRODUTO

No subprocesso de composição, todos os subconjuntos difusos determinados para cada variável de saída são combinados juntos para formar um simples subconjunto difuso para cada variável de saída.

MAX e SOMA são regras de composição. Na composição MAX, a combinação dos subconjuntos de saída é construída pegando o ponto mais discreto máximo sobre todos os subconjuntos difusos determinados para a variável de saída pela regra de inferência. Na composição

SOMA, a combinação dos subconjuntos de saída é feita pegando o maior valor de soma sobre todos os subconjuntos determinados na variável de saída pela regra de inferência. Nota-se que isto pode resultar em valores verdade bem maiores. Por esta razão, a composição SOMA é apenas usada quando for seguida por um método de defuzificação, tal como o método do centróide. Por outro lado, a composição SOMA pode ser considerada como uma normalização e é então um método de propósito geral [Ter95].

Por exemplo, assume-se x = 0.0 e y = 3.2. Na inferência MIN, são determinados os seguintes 4 subgrupos para Z (figura 17):

regra1(z) = {z/10, se z <= 6.8
0.68, se z >= 6.8}

regra2(z) = {0.32, se z <= 6.8
1-z/10 , se > = 6.8

regra3(z) = 0.0

regra4(z) = 0.0

A composição MAX resulta em um subconjunto difuso (figura 3.18):

fuzzy(z) = {0.32, se z <= 3.2 ; z/10, se 3.2 <= z <= 6.8 ; 0.68, se z >= 6.8}

A inferência do produto determina os seguintes 4 subconjuntos difusos para Z:

regra1(z) = 0.68 * Z

regra2(z) = 0.32 - 0.032 * z

regra3(z) = 0.0

regra4(z) = 0.0

A composição SOMA resulta em um subconjunto difuso: fuzzy(z) = 0.32 + 0.036* Z

Algumas vezes ela é útil apenas para examinar os subconjuntos difusos que são os resultados de um processo de composição, mas muitas vezes, este VALOR DIFUSO necessita ser convertido para um simples número - um valor *crisp*. Isto é feito através da defuzificação.

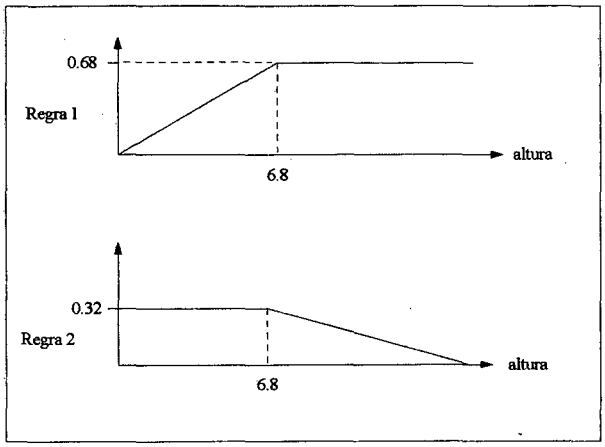

Figura 3.17 Gráficos Relativos às regras 1 e 2.

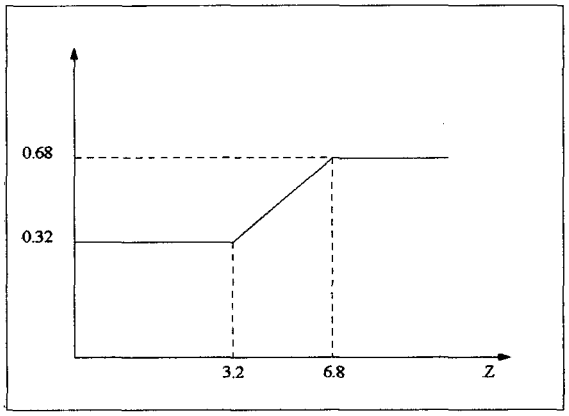

Figura 3.18 Gráfico correspondente à composição MAX.

Utilizando os exemplos anteriores, aplica-se a inferência MAX-MIN e a defuzificação MEDIA DOS MAXIMOS. Isto resulta em um valor *crisp* de 8.4 para Z. Usando a inferência de PRODUTO - SOMA e defuzificando com o CENTRÓIDE, tem-se o valor 5.6 para Z. O cálculo do centróide da função f(x), compreende a divisão do momento da função f(x) pela área da função.

O momento de f(x) é feito através do cálculo da integral de x*f(x) dx. O cálculo da área de f(x), calcula-se a integral de f(x)dx.

No exemplo dado, a área é calculada como integral de 0 a 10 de (0.32 + 0.036*Z)dz, que é: (0.32*10 + 0.018*100) = (3.2 + 1.8) = 5.0 e o momento como integral de 0 a 10 de (0.32*Z + 0.036*Z*Z)dz, que é: (0.16*10*10 + 0.012*10*10*10) = (16+12) = 28. Finalmente, o centróide é 28/5 = 5.6

Algumas vezes, para simplificar o processo de defuzificação usa-se uma função de pertinência de saída representada por uma linha vertical simples. Tal função é chamada *singleton*. Se um *singleton* intercepta o eixo X em apenas um ponto, o cálculo do Centro de Gravidade se reduz a apenas um cálculo médio dos valores dos pontos do eixo X.

CAPÍTULO IV

Sistemas Especialistas Difusos

4.1 Introdução

Os fatos, relações, julgamentos, opiniões e regras de inferência contidos dentro da base de conhecimento do Sistema Especialista, usualmente possuem vários graus de imprecisão e incerteza. Então, é desejável ao Sistema Especialista ser capaz de, como o especialista humano, lidar com inferências de um dado impreciso e heurísticas vagas. Sendo assim, o gerenciamento da incerteza no projeto de um Sistema Especialista é a chave para o sucesso da modelagem do processo de raciocínio, e a utilidade da teoria dos conjuntos difusos para este propósito tem sido e continua sendo extensivamente estudada [Taz87].

As regras empregadas pelos especialistas humanos são de natureza heurística e imprecisa. O uso de valores verdade bivalentes ou requerimento de uma satisfação exata da cláusula condicional IF para a avaliação de uma regra sempre parece anormal no contexto do raciocínio humano.

4.2 Justificativa do uso de Sistemas Especialistas Difusos

Sistemas baseados em lógica difusa são bem menos complexos que os tradicionais. Eles podem ser facilmente compreendidos, além de isolarem e resolverem problemas de maneira bem mais rápida, e com poucas regras. A média de tempo entre as falhas geralmente se aproxima de um valor considerado próximo ao desejável.

Os principais benefícios derivados do uso de modelos difusos em sistemas especialistas são: (i) a capacidade de modelar problemas altamente complexos; (ii) melhoria da modelagem cognitiva dos sistemas especialistas; (iii) habilidade de modelar sistemas envolvendo vários especialistas; (iv) redução da complexidade do modelo; (v) melhoria da capacidade de manipulação da "incerteza" e da "possibilidade" [Cox94].

Os sistemas convencionais enfrentam sérias dificuldades no que diz respeito a problemas não lineares e computacionalmente complexos. Os sistemas difusos, porém, utilizam regras difusas, as quais diminuem sensivelmente a complexidade de um problema. Sistemas baseados em regras difusas têm a execução bem mais rápida do que os sistemas convencionais e requerem poucas regras.

Para muitos engenheiros do conhecimento, um benefício muito importante do uso dos sistemas difusos é a habilidade de codificar o conhecimento diretamente em uma forma muito próxima da maneira como os especialistas pensam a respeito de um problema. Os sistemas convencionais forçam os especialistas a fornecerem valores SIM/NÃO para as variáveis, levando a uma multiplicação desnecessária de regras. Além disso, esta abordagem convencional inibe a capacidade do especialista em articular uma solução para um problema complexo. Já que o sistema difuso captura o conhecimento de maneira muito próxima da realidade, o processo de aquisição do conhecimento é mais fácil e menos ambíguo[Tur91].

Na literatura correspondente aos sistemas especialistas, há sempre uma tendência a transmitir a idéia de que apenas um especialista ou, que todos os especialistas em um campo concordam em todos os aspectos. No mundo real, este é um caso raro. Os sistemas difusos são bastante adequados para representar a cooperação entre vários especialistas, bem como seus conflitos.

Os modelos difusos requerem menos regras que os sistemas convencionais e estas regras estão bem próximas da maneira como o conhecimento é expresso em linguagem natural. Isto traz dois importantes benefícios. Primeiro, um modelo pode ser modificado com pouca margem de erros.

Segundo, a relativa simplicidade de um modelo difuso significa que os problemas lógicos e estruturais podem ser localizados e solucionados em pouco tempo.

A característica básica de um Sistema Especialista Difuso é a descrição lingüística do raciocínio humano. Isto ocorre através do uso das variáveis lingüísticas.

As variáveis lingüísticas são variáveis cujos valores não são números, mas palavras ou sentenças, ao invés de números. Sendo assim, as caracterizações lingüísticas são, em geral, menos específicas que as caracterizações numéricas [Zim92].

4.3 A Modelagem da Incerteza nos Sistemas Especialistas

Há três situações básicas de imprecisão (incerteza) nos Sistemas Especialistas que não são consideradas por técnicas tradicionais [Zim92]:

1) A difusão de antecedentes e/ou consequentes em regras da forma:

a) SE x é A, ENTÃO y é B.

b) SE x é A, ENTÃO y é B, com FC (fator de certeza) = α

onde, o antecedente X é A, o consequente Y é B, são proposições difusas e o FC é um valor numérico. Por exemplo, SE x é pequeno, ENTÃO y é grande, com FC = 0.8.

2) O relacionamento parcial entre o antecedente de uma regra e um fato fornecido pelo usuário:

fato: X é A*

regra: Se X é A, Então y é B com FC = β

Como exemplo, tem-se "Se x é pequeno, Então y é grande com FC = 0.8", onde X significa a altura humana. Caso o valor fornecido a X pelo usuário fosse 1.60m, haveria relacionamento parcial da regra, devido a pertinência de 1.60 ao conjunto difuso dos valores "Altura Pequena".

Este tipo de situação é evitada em sistemas especialistas tradicionais. O relacionamento parcial não pode ser tratado dentro da abordagem da lógica de duplo valor.

3) A presença de quantificadores difusos no antecedente e/ou consequente de uma regra:

Os quantificadores difusos (maioria, muito, pouco, etc.) estão frequentemente presentes no conhecimento humano. Um exemplo para elucidar o mecanismo de "tradução das disposições"(proposição com quantificadores difusos) em regras é exemplificado a seguir:

disposição d = os estudantes são jovens. - que pode ser interpretado por:

proposição p = a maioria dos estudantes é jovem. - que por sua vez, pode ser expressa como uma regra ou equivalente como proposição condicional.

regra = Se x é um estudante, Então é provável que seja jovem. - onde a probabilidade difusa "provável" tem o mesmo significado expresso como um subconjunto difuso de intervalo unitário representado pelo quantificador difuso "MAIORIA".

Essas três situações comprometem as conclusões oriundas de tratamento tradicional. Este tratamento manipula fatos e regras difusas na realidade, como sendo não difusos. Assim, as conclusões têm sua validade aberta a questionamentos.

4.4 Principais Razões para a Utilização da Teoria dos Conjuntos Difusos em Sistemas Especialistas

Há três principais razões para se utilizar a teoria dos conjuntos difusos nos Sistemas Especialistas [Ter95].

a) As *interfaces* do Sistema Especialista tanto no lado do especialista quanto no lado do usuário, lidam com seres humanos. Então, a comunicação de uma maneira "natural" tem de ser mais apropriada, e "natural" significa, na linguagem do especialista ou do usuário. Isto sugere o uso de variáveis lingüísticas.

b) A base de conhecimento de um Sistema Especialista é um repositório do conhecimento humano e, sendo este impreciso por natureza, é sempre comum que a base de conhecimento de um Sistema Especialista seja uma coleção de regras e fatos que, em sua maioria, não está totalmente correta e consistente. O armazenamento deste conhecimento difuso e incerto pelo uso de conjuntos difusos, parece muito mais apropriado que o uso de conceitos exatos.

c) O "gerenciamento de incerteza" desempenha um papel particularmente importante. A incerteza da informação nas bases de conhecimento induz a incerteza nas conclusões e, então, a máquina de inferência tem de ser equipada com capacidades computacionais. Tais capacidades devem transmitir a incerteza das premissas para as conclusões e associar a conclusão com algumas medidas de incerteza que são compreensivelmente e propriamente interpretadas pelo usuário final.

d) Em um ambiente preciso, as regras de produção são adequadas para representação do conhecimento, enquanto que tal não é possível em um ambiente difuso. Uma forma de lidar com a imprecisão é usar as regras de produção difusas, onde a parte condicional e/ou, a parte de conclusões contém variáveis lingüísticas [Kli92].

4.5 As Vantagens da Utilização da Lógica Difusa em Sistemas Especialistas

1) Tratamento de Proposições: em lógica bivaloradas, uma proposição p ou é V ou F. Em lógica polivalorada, ela pode ser V ou F ou ter um valor verdade intermediário que pode ser um elemento do conjunto finito ou infinito U de valores de verdade. Na lógica difusa, os valores de verdade podem variar sobre subconjuntos difusos U.

2) Tratamento de Predicados: ao contrário do que ocorre na lógica de duplo valor, onde os predicados devem ter tratamento clássico, aqui pode-se assumir a forma rígida (mortal, pai de, sempre, etc.) ou, mais genericamente, a forma difusa (doente, cansado, grande, alto, etc.).

3) Tratamento de Quantificadores: a quantificação de expressões em lógica de duplo valor permite apenas a aplicação dos termos "todo" e "alguns". Na lógica difusa, a generalidade é permitida na utilização de expressões como maioria, muitos, etc. Assim, os quantificadores expressam de forma imprecisa a cardinalidade dos conjuntos difusos através da caracterização dos mesmos como predicados difusos de segunda ordem.

4) Tratamento dos Modificadores de Predicado: os modificadores (mais ou menos, extremamente, etc.) são passíveis de representação em lógica difusa. Deste tratamento surgem sistemas que consideram variáveis lingüísticas, ou seja, variáveis cujos valores são palavras ou sentenças em linguagem natural.

5) Qualificação de Proposições: para classificar uma proposição p em lógica de duplo valor, utilizam-se termos V ou F, operadores modais como possível ou necessário e operadores intensionais, como sabe-se, acredita-se, etc. Em lógica difusa, há três modos de qualificação:
a) Qualificação verdade: p é t, na qual t é um valor V difuso.

b) Qualificação probabilística: p é λ, na qual λ é uma probabilidade difusa.

c) Qualificação possibilística: p é π, na qual π é uma possibilidade difusa (ex: muito provável, etc.).

4.6 Exemplos de Sistemas Especialistas Difusos

Alguns exemplos de Sistemas Especialistas Difusos serão descritos à seguir. Tais sistemas foram apresentados no II Congresso Mundial de Sistemas Especialistas (Portugal, 1994):

- ***AFRS:*** Sistema Adaptativo Difuso Baseado em Regras;
- ***ESED:*** Sistema Especialista em Desenvolvimento Econômico;
- ***ExTRA:*** Sistema Especialista para Alocação Tática dos Componentes do Ativo;
- ***FOREX:*** Sistema Especialista para Previsão de Tendências de Taxa de Câmbio entre o *yen* e o dólar;
- ***RBO-GIDIA:*** *Shell* para desenvolvimento de Sistemas Especialistas Difusos utilizando o conceito de "objetos difusos";

4.6.1 AFRS - *Adaptative Fuzzy Rule-Based System* (Depto. de Ciência da Computação da Universidade de Sogang - Coreia)

A maioria dos sistemas baseados em conhecimento têm por base regras. Tais sistemas são construídos por regras iniciais que são adquiridas a partir de especialistas. A aquisição deste conhecimento e a estruturação do mesmo são operações complexas. Como solução para isto, criou-se o AFRS, que tem por objetivo encontrar o relacionamento entre dados fornecidos pelo mundo real e gerar regras baseadas nestes relacionamentos.

Algoritmos genéticos foram usados como método de criação das regras iniciais. Os números difusos foram utilizados para selecionar uma regra com maior grau de pertinência entre as várias regras que satisfazem uma dada condição de entrada. O diagrama de blocos do AFRS é ilustrado na figura 4.1.

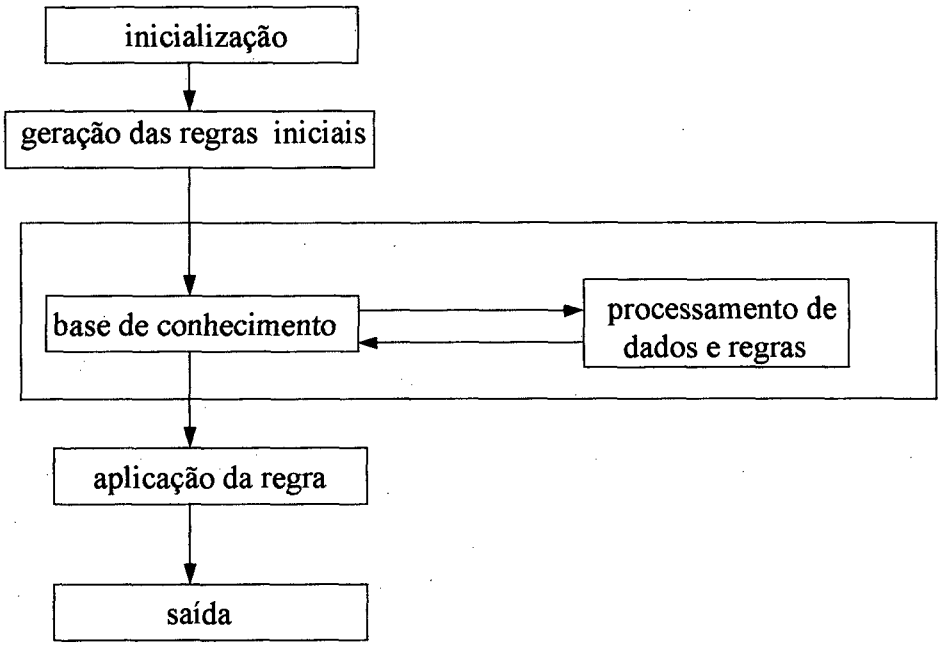

Figura 4.1 - Diagrama de Blocos do AFRS.

A forma sintática das regras usadas segue o seguinte modelo:

IF (A (sub 1), A(sub 2),...(A(sub n)) (é um elemento de) R (sup m)

THEN B(sub i) = f(A(sub 1), A(sub 2),... A(sub n))

onde, A (sub 1), A (sub 2), ..., A(sub n)) são valores difusos de um atributo, n é o número de atributos, e R significa um espaço m-dimensional. B(sub i) é uma ação que depende dos valores dos atributos difusos dos antecedentes.

Os parâmetros e escopos das regras são alterados durante o processo de aprendizagem. Tal processo é dividido em duas fases: expansão e fragmentação.

A fase de expansão ocorre quando uma dada entrada não está contida no escopo das regras existentes. Esta fase está dividida em: (i) a expansão de algumas regras; (ii) ajuste dos parâmetros das regras.

A fase de fragmentação ocorre quando uma dada entrada está contida no conjunto de regras existentes e viola as condições dadas quando uma das regras é aplicada. Uma separação é feita dividindo as regras em pequenas partes. As regras representando estas partes herdam as propriedades das regras anteriores. Após o processo de aprendizagem, os parâmetros e o escopo das regras que sofreram fragmentação podem ser alterados de acordo com os dados de entrada.

Quando muitas regras satisfazem uma dada condição de entrada, AFRS seleciona uma regra com mais alto grau de pertinência usando os números difusos para solucionar o conflito.

A figura 4.2 mostra o escopo de uma regra através dos pontos P_1 e P_2. Neste exemplo, se uma regra contem um certo ponto P_i, o grau de pertinência é obtido usando um número difuso triangular. Os pontos com grau de pertinência acima de 0.5 são aceitáveis. A linha mais grossa corresponde aos pontos aceitáveis.

4.6.2 ESED - *Expert System for Ecomonic Development* (Depto. de Ciência da Computação da Universidade da Cailfórnia - Berkeley , Estados Unidos)

O método tradicional para a modelagem econômica de prognóstico-prescrição, envolve a criação de modelos baseados em dados empíricos obtidos à partir de investigações econômicas passadas. O ESED foi desenvolvido com o objetivo de criar uma base de regras de prescrições e prognósticos econômicos. A implementação foi feita utilizando as linguagens FRILL e FPROLOG Para que este sistema fosse tolerante às imprecisões e flexível, seu desenvolvimento baseou-se na

lógica difusa. Exemplos de proposições difusas em economia são bastante comuns: "inflação é moderada"; "desemprego é baixo"; etc. Aqui, "moderado" e "baixo" são predicados difusos; "inflação" e "desemprego" são variáveis.

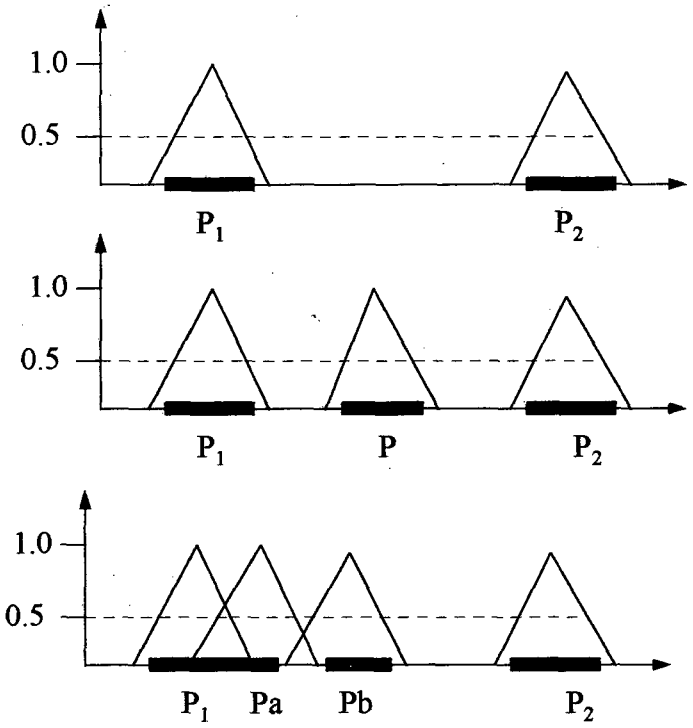

Figura 4.2 Escopo de uma Regra Usando P_1 e P_2.

A figura 4.3 ilustra a estrutura do ESED.

A base de conhecimento é divida em módulo de mapeamento, módulo fuzificador, módulo defuzificador e módulo de interpolação e uma base de regras difusas.

As regras difusas são geradas e cada variável difusa é mapeada para um número difuso, por exemplo: baixa, média e alta para a variável INFLAÇÃO. Se uma variável de entrada tem um valor *crisp*, ela é fuzificada pelo módulo fuzificador. Se uma entrada tem um valor difuso, seu valor é interpolado usando os conjuntos difusos vizinhos. A máquina de inferência manipula e raciocina com regras e fatos. Fatos fornecidos e gerados são armazenados na memória de trabalho para uso futuro.

A *interface* do usuário permite que o usuário interrogue a base de conhecimento. Os módulos de explicação fornecem uma justificativa para as respostas que o sistema fornece à uma pergunta, através da listagem dos fatos e regras que foram disparadas como resultado para uma referida pergunta.

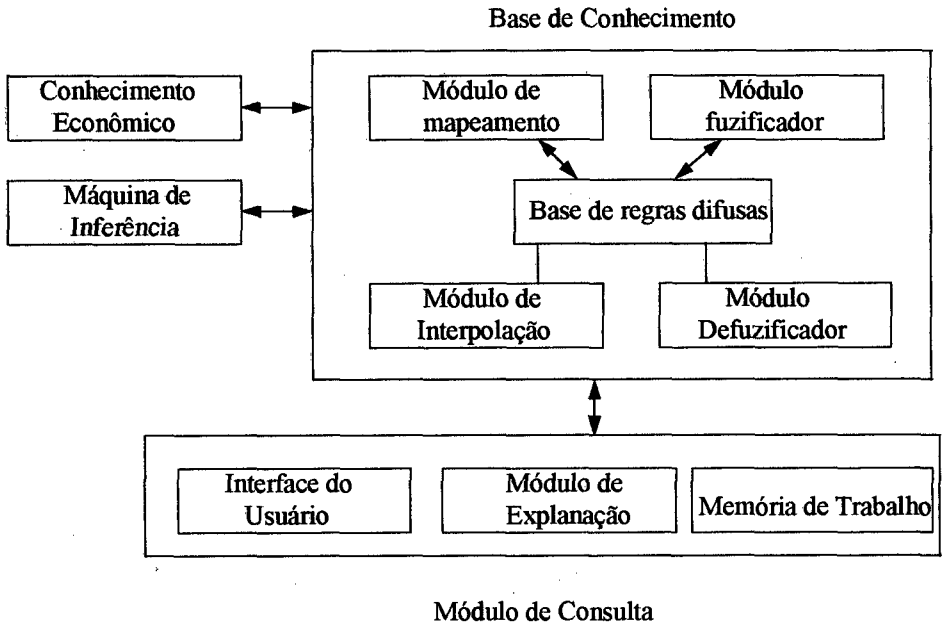

Figura 4.3 - Estrutura do Sistema ESED.

Um exemplo de raciocínio difuso utilizando lógica difusa no ESED é ilustrado na figura 4.4, através da função de pertinência para inflação, baixa, moderada e alta; e as respectivas taxas de desemprego.

Uma inflação de 1% ou 12% não pode racionalmente ser considerada como moderada. Entretanto, os dois valores pertencem a um conjunto de "inflações moderadas" com grau de pertinência zero. Por outro lado, uma inflação de 4% pode ser definitivamente considerada como moderada.

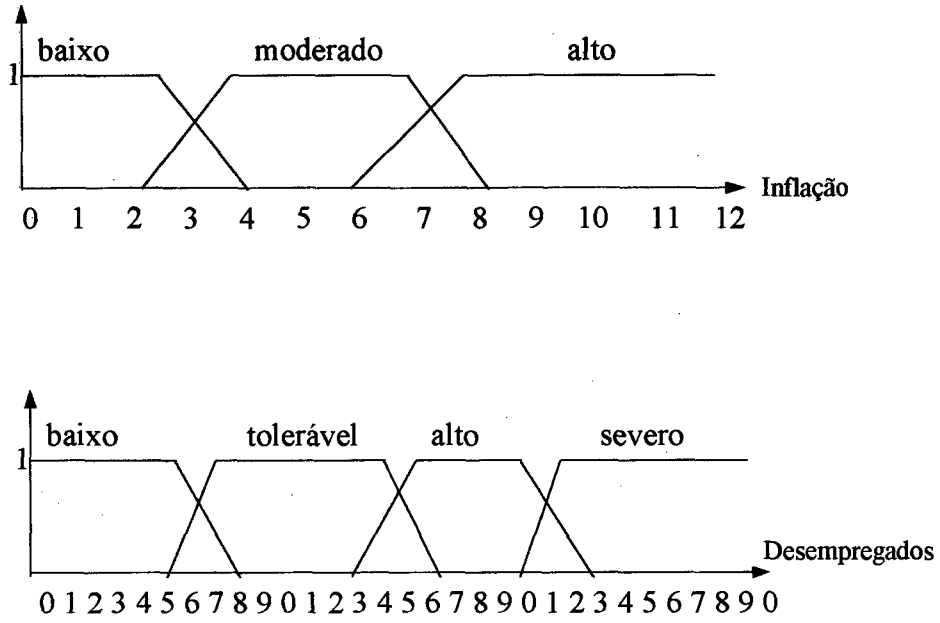

Figura 4.4 - Representação de Funções de Pertinência do ESED.

4.6.3 - ExTRA - *Expert system for Tactical Re-Allocation* (Depto. de Sistemas de Informação, Universidade de Amsterdam, Holanda)

ExTRA é um sistema especialista para alocação de componentes táticos do ativo. Ele possui três componentes, cada um dos quais realiza uma tarefa específica: (i) componente baseado em diagnósticos, interpreta as condições de investimentos macroeconômicos, (ii) componente baseado em regras difusas, prevê o retorno de mercadorias com base nas informações do componente anterior; (iii) componente de otimização baseado na variação média do portfólio faz a seleção de modelos específicos de ajuste.

A figura 4.5 mostra as tarefas genéricas da alocação de componentes táticos do ativo.

O ExTRA foi projetado com o objetivo de solucionar os problemas de cada uma destas tarefas de maneira interativa e fornecer um suporte inteligente para tomadas de decisão.

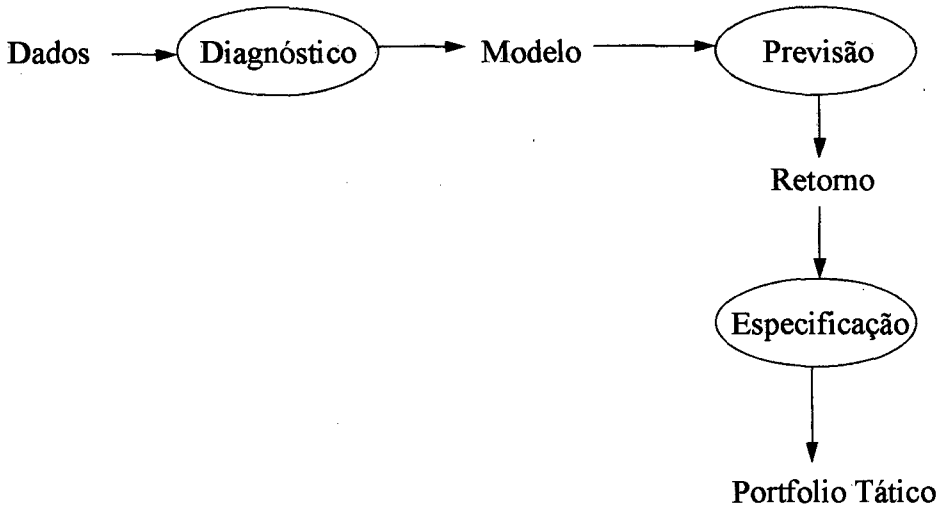

Figura 4.5 - Tarefas genéricas da alocação de componentes táticos do ativo.

O princípio básico do esquema de previsão está relacionado com uma informação fundamental sobre condições de investimento macroeconômico e avaliação de mercado para o retorno do excesso de estoque de mercado.

ExTRA representa o conhecimento sobre conceitos que fazem a previsão do estoque de mercado em forma de proposições difusas. O conhecimento para relatar estes conceitos é representado como regras difusas. O processamento é controlado em sequência por unidades de inferência difusas (FIU), cada uma das quais relaciona-se com uma variável em particular. A seguir tem-se o exemplo de uma regra da FIU responsável pelo retorno do excesso.

IF

Avaliação-de-Mercado = Baixo AND Índice-de-investimento = Alto

THEN

Retorno-de-excesso = Alto

ExTRA foi implementado usando três ferramentas que rodam em ambiente "Microsoft Windows" .O componente de otimização foi implementado em "Mathematica", o componente relativo à lógica difusa foi feito em "CubiCalc" e o componente de diagnóstico em "Kappa-PC".

4.6.4 - FOREX - *FOReign exchange trade support EXpert system* (Laboratório para Pesquisas Internacionais em Lógica Difusa - Japão)

O FOREX foi desenvolvido com o objetivo de prever as variações de câmbio entre o yen e o dólar. As previsões feitas por ele são baseadas não apenas em representação de valores numéricos, mas também em itens que são fornecidos ao sistema na forma de expressões em linguagem natural.

Durante o processo de aquisição do conhecimento, vários analistas econômicos foram entrevistados, e chegou-se a conclusão de que as relações entre os itens econômicos podem ser expressas por variáveis lingüísticas tais como: "IF taxa-FF se torna alta THEN termos-de-interesse-a-curto-prazo serão altos". Esta expressão do conhecimento econômico possui uma semelhança com a linguagem natural, então, para expressar este tipo de conhecimento, usou-se regras de produção difusas. Devido à natureza do grau de relacionamento entre as partes antecedentes e consequentes da regra, e devido ao efeito sinérgico entre os diferentes itens da parte antecedente, os métodos usuais de inferência difusa não foram considerados adequados para este sistema.

O grau de relacionamento pode ser definido como sendo "o mais alto grau para o qual itens do antecedente de uma regra satisfazem os predicados das condições, o mais alto grau para o qual o resultado na porção consequente da regra satisfaz os predicados das conclusões". Esta é a relação entre o antecedente e o consequente da regra.

Quando há muitos itens no antecedente de uma regra, o valor verdade total da condição é geralmente obtido usando uma combinação de operadores **t-norms**. Mas estes operadores não são adequados para representar o efeito sinérgico usado no FOREX. A primeira proposta para este problema foi o uso de um operador "and". Este operador é definido usando muitos parâmetros. Porém o operador "and" comum não atendia às necessidades, sendo então proposto um novo tipo de operador "and" que solucionasse o problema do FOREX.

O operador "and" é usado para combinar os valores verdade dos antecedentes de uma regra, e pode ser definido como: "and" : $[0,1] * [0,1] \rightarrow [0,1]$

Quando todos os valores são "meio alto", (isto é, quando cada item do antecedente satisfaz o predicado com um grau "meio alto"), a saída do operador "and" tem que ser maior que o operador **t-norm** usual. Para outros valores de entrada, o "and" é o operador **t-norm** usual.

O algoritmo para definir o operador "and" consta de três funções: (i) função básica - combinações básicas do operador (t-norm); (ii) função sinérgica - expressa as propriedades sinérgicas; (iii) função de definição de área - define a área onde o efeito sinérgico é solicitado.

Usando estas funções, o operador "and" é definido como a média dos pesos das funções básica e sinérgica, usando a função de definição de área. A figura 4.6 mostra o algoritmo básico usado para construir o operador "and".

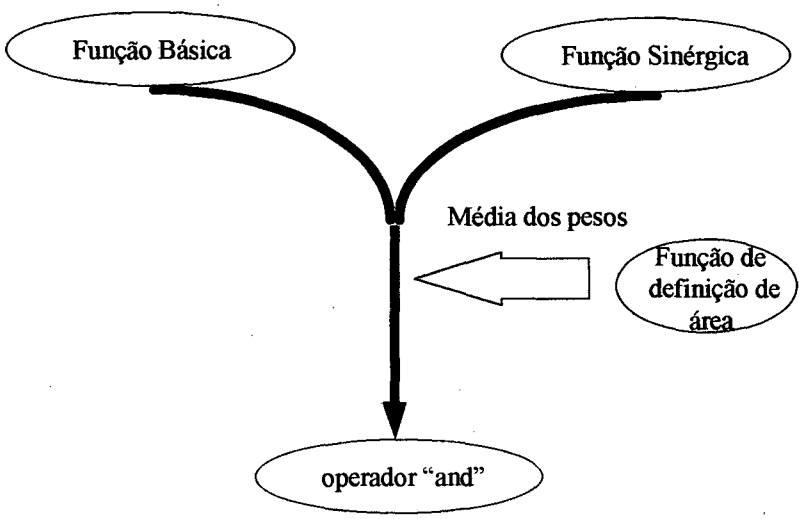

Figura 4.6 - Algoritmo básico usado para construir o operador "and".

4.6.5 - RBO - GIDIA (Depto. de Ciência da Computação da Universidade de Buenos Aires, Argentina)

O projeto RBO-GIDIA tem como objetivo construir uma *shell* geradora de sistemas especialistas, de acordo com a proposta KP-GIDIA. Esta ferramenta é baseada em objetos e lida com o raciocínio aproximado. Uma importante característica é a possibilidade de definir objetos difusos e realizar *fuzzy matching* entre os objetos.

O KP - GIDIA é um projeto que consiste em desenvolver ambientes de programação inteligentes para servir como base para a geração de um *driver* de inferência a ser usado na construção de sistemas especialistas. O ambiente KP-GIDIA é constituído por uma série de ferramentas fáceis de serem usadas pelo usuário, as quais podem ser adaptadas à área em questão.

Para ser o núcleo central do sistema, um *driver* baseado em objetos foi projetado e chamado RBO-GIDIA (*Object Based Reasoning Feature*). Esta ferramenta é o eixo sobre o qual as

inferências são feitas. Embora a teoria da Programação Orientada a Objeto já esteja solidificada, pensou-se em ampliá-la com conceitos de lógica difusa, usando-se o conceito de objetos difusos.

Se um usuário pergunta sobre fatos em R, geralmente, ele obtem uma resposta correta. Mas pode haver uma situação onde um fato particular não unifica com nenhum objeto da base de conhecimento. Ele pode ter um atributo que não coincida com os valores armazenados na base. Neste caso, a unificação com o objeto mais similar já é útil. Para isto, é necessária uma função capaz de pesquisar esta similaridade entre os objetos.

Formalmente, uma função de *fuzzy-matching* M:H(R)*J(R)--L[0,1], onde H(R) e J(R) são os conjuntos de fatos (O..L) que são possíveis e aqueles que são inferíveis em R respectivamente. Então, na presença de uma pergunta, a resposta seria : R = {O'.. L': M(O..L, O'..L') = m}

O problema é definir estas funções. Uma base poderia definir as funções de *fuzzy matching* a partir do índice de similaridade entre os valores dos atributos, e o peso para cada um destes atributos.

Esta ferramenta envolve a implementação do raciocínio difuso e funções de *fuzzy matching* entre os objetos em PROLOG através da definição de atributos difusos pela função reservada "f". Por exemplo: jovem = f(idade), de maneira que se possa explicar que jovem é um atributo difuso necessário para avaliar a idade.

Em RBO_GIDIA, pode-se escrever :

:- create_node(canário).
:- create_node(twity).
:- create_node(pássaro).

O conhecimento é representado por:

'@'(canário, arc, comida).
'@'(canário, arc, voar).
'@'(twity, arc, idade).
'@'(twity, arc, jovem).
'@'(twity, is_a, canário).
'@'(canário, is_a, pássaro).
'@'(twity, comida, laranjas).

O usuário tem que escrever a função difusa descrevendo quando alguém é jovem e é caracterizada pela sua função de pertinência:

fuzzy(jovem, idade, 1):- Idade >=0, Idade < 20.
fuzzy(jovem, idade, 0.8):- Idade >= 20, Idade < 30.

Para a criação de nós, RBO-GIDIA tem:

```
create_node(X,L):-
        assert('@nodo'(X)).
```

```
create_node(X,L):-
        assert('@nodo'(X)),
        eval(X..L).
```

O **ev_fuzzy** é outra função reservada que retorna o grau de pertinência, e chama a função **fuzzy-translate**. O predicado é traduzido em termos de grau de pertinência.

```
ev_fuzzy(FUZZY):-[!(FUZZY = [F, (O..L), f(A)], ev (O..A, N1))!],
        fuzzy(L, N1, N), !, nl, fuzzy_translate (O, L, N).
```

O predicado PROLOG *fuzzy-match* faz um *fuzzy matching* entre dois objetos e retorna a escala de similaridade:

```
fuzzy_match (Objeto1, Objeto2, Escale):-
        atributos(Objeto1, Lista1),
        atributos(Objeto2, Lista2),
        heurística(H), estratégia (E),
        como(Lista1, Lista2, H, E, Escala).
```

onde "heurística" e "estratégia" contem as heurísticas e estratégias para solucionar o problema.

CAPÍTULO V

Estudo de Ferramentas para Desenvolvimento de Sistemas Especialistas Difusos

5.1 Introdução

Inicialmente, cada sistema especialista era criado a partir do nada, em geral em LISP. Mas, depois de vários sistemas terem sido desenvolvidos, ficou claro que esses sistemas tinham muito em comum, só se diferenciavam no que diz respeito ao conhecimento armazenado. Baseado nesta semelhança entre os sistemas, criaram-se as *Shells*.

Uma *Shell* consiste em um sistema com todas as partes constituintes de um sistema especialista, com exceção da base de conhecimento. Assim, cabe ao engenheiro do conhecimento colocar no sistema apenas o conhecimento adquirido junto ao especialista, já que a *Shell* possui o mecanismo de inferência para manipulá-las. O objetivo principal de uma *Shell* é permitir ao próprio especialista entrar com suas regras [Ric91] [Wat86].

5.2 CLIPS

A origem de CLIPS (*C Language Integrated Production System*) [Lbj93], 1984 no *Johnson Space Center - NASA* (EUA). Nesta época, a Seção de Inteligência Artificial, atualmente, *Software Technology Branch,* vinha desenvolvendo dezenas de protótipos de aplicações de sistemas especialistas. Entretanto, apesar da extensiva demonstração do potencial destes sistemas, poucas destas aplicações tinham uso regular. Esta falha em fornecer uma tecnologia para sistemas especialistas dentro do esquema de computação operacional da *NASA* poderia ser devido ao uso de

LISP como a linguagem base para tais sistemas. Em particular, três problemas impediram o uso de LISP na *NASA*: a baixa viabilidade do LISP em uma ampla variedade de computadores convencionais, o alto custo das ferramentas LISP e do *hardware* adequado e pouca integração do LISP com outras linguagens.

O uso de ferramentas baseadas em linguagem C tinha o custo ainda muito alto, e a maioria era restrita a uma pequena variedade de computadores, além disso, o tempo de desenvolvimento do projeto era desencorajador. Para sanar todas estas necessidades de uma maneira barata e rápida, a Seção de Inteligência Artificial desenvolveu então, sua própria ferramenta, baseada em linguagem C, lançada em 1985, a linguagem CLIPS.

A sintaxe do CLIPS é muito parecida com a linguagem ART, usada em várias ferramentas de construção de sistemas especialistas.

A metodologia de representação do conhecimento em CLIPS é o encadeamento de regras para frente, baseado no algoritmo de **Rete.** As programações procedurais e orientadas-a-objeto são os paradigmas disponíveis.

Devido à sua portabilidade, extensibilidade, capacidade e baixo custo, o CLIPS tem tido grande aceitação em órgãos do governo, indústrias e universidades. O desenvolvimento do CLIPS tem ajudado a melhorar a difusão da tecnologia de sistemas especialistas entre as instituições privadas e usuários em geral [Lbj93].

Quando se usa um sistema especialista, dois tipos de integração são importantes: embutir a ferramenta em um outro sistema e chamar funções externas de dentro da ferramenta. CLIPS foi projetado para suportar os dois tipos de integração.

Usando CLIPS como uma aplicação embutida, há uma fácil integração deste com os sistemas existentes. Isto é útil em casos onde o sistema especialista é uma pequena parte de uma tarefa bastante ampla ou necessita compartilhar dados com outras funções. Nestas situações, CLIPS pode ser chamado como uma subrotina e a informação pode passar de ou para o CLIPS.

Pode também ser útil chamar funções externas enquanto se está executando uma aplicação CLIPS ou a partir de uma *interface* de alto nível. As variáveis ou valores literais do CLIPS podem ser enviados para uma função externa, e estas podem retornar valores para o CLIPS. A fácil adição de funções externas permite que o CLIPS seja estendido ou customizado em qualquer forma (ou seja, para qualquer aplicação).

CLIPS fornece paradigmas heurístico e procedural para a representação do conhecimento.

5.2.1 Conhecimento Heurístico - Regras

A Regra consiste em um dos principais métodos de representação do conhecimento em CLIPS. Regras são usadas para a representação heurística, ou métodos de inferência, os quais especificam um conjunto de ações a serem realizadas para uma dada situação. Quem desenvolve um sistema especialista define um conjunto de regras que trabalham coletivamente para solucionar o problema.

Em CLIPS, as condições de uma regra são satisfeitas baseadas na existência ou não de fatos especificados na lista de fatos ou nas instâncias das classes definidas pelo usuário, na lista de instâncias. Um tipo de condição que pode ser especificada é um padrão. Os padrões consistem em um conjunto de restrições que são usadas para determinar quais os fatos ou objetos que satisfazem a condição especificada pelo padrão. O processo de unificação dos fatos e objetos com os padrões é

chamado *pattern matching*. O CLIPS fornece uma máquina de inferência que automaticamente compara os padrões com as listas de fatos e instâncias e determina qual regra deve ser aplicada.

O consequente de uma regra é o conjunto de ações a serem executadas se a regra for aplicada. As ações das regras aplicáveis são executadas quando a máquina de inferência do CLIPS é instruída a iniciar a execução destas regras. Se mais que uma regra é aplicável, a máquina de inferência usa a estratégia de resolução de conflitos para selecionar a regra a ser executada e então, a máquina de inferência seleciona a mesma e executa suas ações. Este processo continua até não restar regras aplicáveis.

5.2.1.1 Algoritmo de RETE

As linguagens baseadas em regras, tais como CLIPS, ART, OPS5 e OPS83 usam um algoritmo bastante eficiente para unificação dos padrões nas regras, a fim de determinar quais regras têm seus antecedentes satisfeitos. Este algoritmo é chamado de **Algoritmo de Unificação de Padrões de RETE** [Gia89]. A figura 5.1 ilustra o problema tratado por este algoritmo.

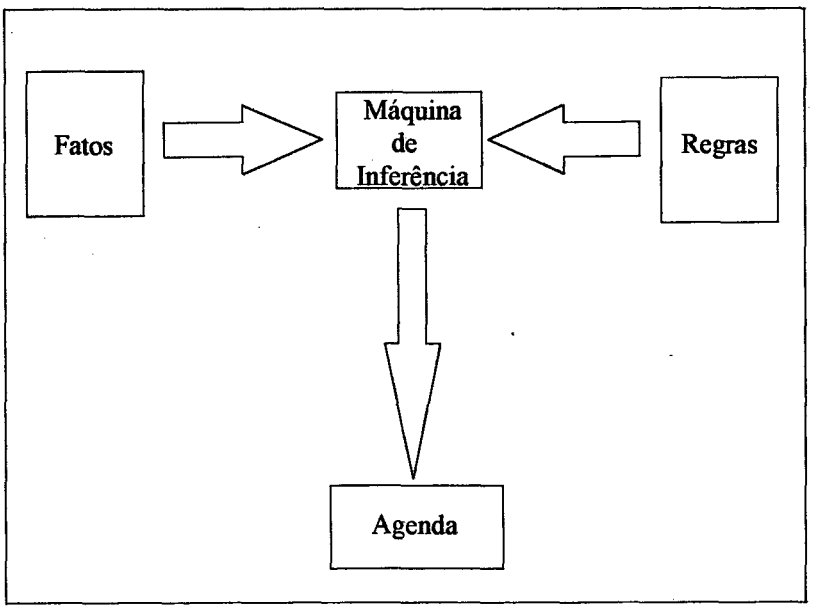

Figura 5.1 Unificação de Padrões: Fatos e Regras.

Se o processo de unificação tiver que acontecer apenas uma vez, então a solução para o problema é simples. A máquina de inferência pode examinar cada regra e, então, procurar o conjunto de fatos que determinam se os padrões da regra serão satisfeitos. Se os padrões das regras são satisfeitos, então, a regra pode ser colocada na agenda. A figura 5.2 ilustra as regras procurando pelos fatos necessários para unificar suas condições.

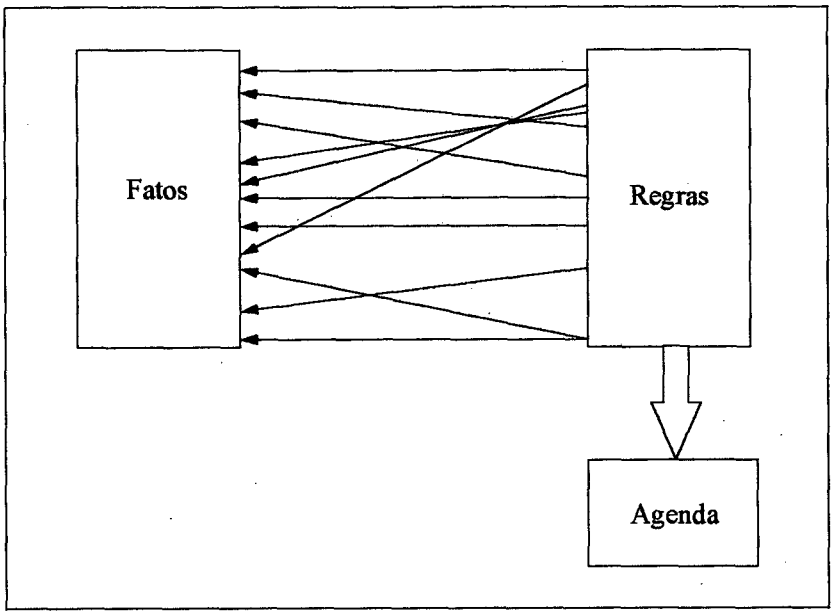

Figura 5.2 Regras à Procura de Fatos.

Em linguagens baseadas em regras, entretanto, o processo de unificação ocorre repetidamente. Normalmente, a lista de fatos será modificada durante cada ciclo de execução. Novos fatos podem ser acrescentados à lista de fatos ou fatos antigos podem ser removidos desta lista. Estas mudanças podem fazer com que padrões que não foram satisfeitos anteriormente sejam satisfeitos ou que padrões satisfeitos anteriormente não sejam satisfeitos. O problema de unificação se torna um processo contínuo. Durante cada ciclo, os fatos são acrescentados e removidos, o conjunto de regras que são satisfeitas devem ser mantidas e atualizadas [Bro85].

Tendo a máquina de inferência checado cada regra para direcionar a busca dos fatos, após cada ciclo de execução, é fornecida uma técnica para a solução deste problema. A primeira desvantagem desta técnica é que ela é muito lenta. Muitos sistemas especialistas baseados em regras

possuem a propriedade chamada de **redundância temporal**. Tipicamente, as ações de uma regra irão mudar apenas alguns fatos na lista de fatos. Isto é, os fatos no sistema especialista mudam lentamente. Cada ciclo de execução pode tratar apenas uma pequena porcentagem de fatos (acrescentar ou remover) e então, apenas uma pequena porcentagem de regras é afetada pelas mudanças na lista de fatos. Uma vez que as regras guiam a busca pelos fatos necessários, um tempo desnecessário é gasto, uma vez que as regras irão percorrer toda a lista de fatos a cada ciclo. A ineficiência deste método é ilustrada na figura 5.3. A área sombreada representa as mudanças que são feitas na lista de fatos. Uma busca desnecessária pode ser refeita para relembrar o que já foi unificado ciclo a ciclo e computando apenas as mudanças necessárias para a remoção e deleção de fatos (figura 5.4). Concluí-se então, que as regras permanecem estáticas e somente os fatos mudam. Sendo assim, os fatos poderiam procurar as regras e não as regras procurarem os fatos.

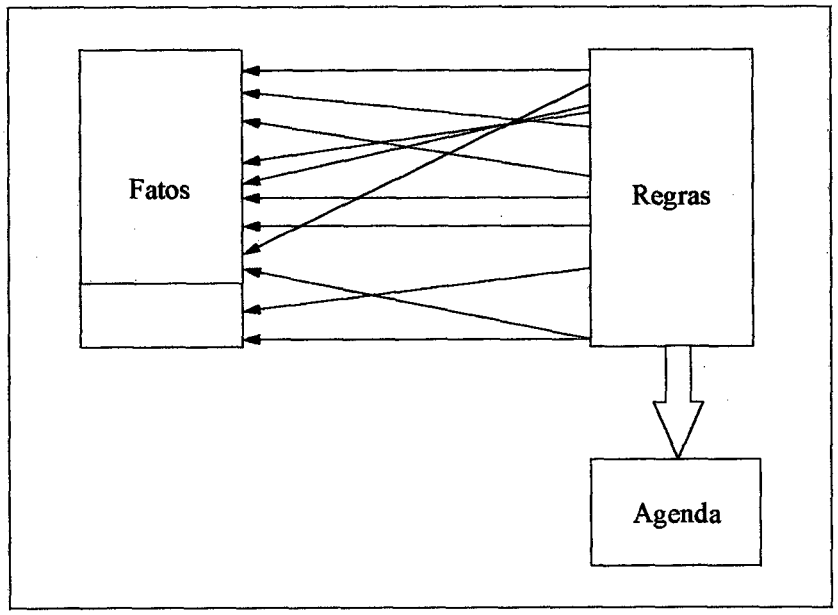

Figura 5.3 Computação Desnecessária Quando as Regras Procuram os Fatos.

O algoritmo de unificação de padrões de RETE foi projetado para tirar vantagem da redundância temporal que existe em um sistema especialista baseado em regras [For85]. Ele faz isto salvando o estado do processo de unificação de ciclo por ciclo e recalculando as mudanças de um determinado estado apenas para aquelas que ocorreram na lista de fatos. Isto é, se um conjunto de padrões encontra dois dos três fatos solicitados em um ciclo, uma checagem não precisa ser feita no próximo ciclo para dois fatos que já foram encontrados. Apenas o terceiro fato é de interesse. O

estado do processo de unificação é atualizado apenas se fatos são acrescentados ou removidos. Se o número de fatos acrescentados e removidos é pequeno quando comparado ao total de números de fatos e padrões, então, o processo de unificação será rápido. Na pior das hipóteses, se nenhum dos fatos é alterado, então, o processo de unificação trabalha unificando todos os fatos com todos os padrões.

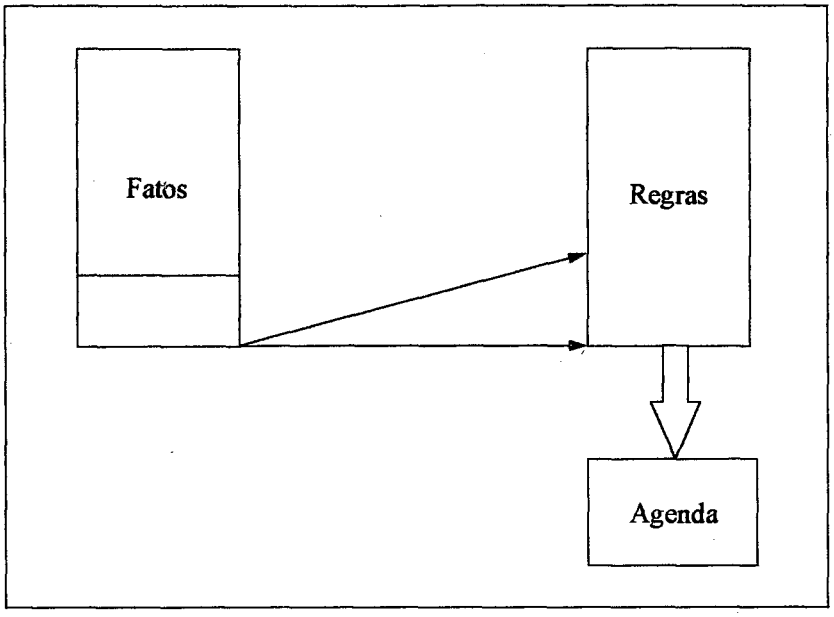

Figura 5.4 Fatos em Busca das Regras.

Se apenas as atualizações dos fatos são processadas na lista de fatos, então, cada regra tem que relembrar quais os fatos que foram unificados às mesmas. Isto é, se um novo fato é unificado ao terceiro padrão de uma regra, então, a informação sobre as unificações para os dois primeiros padrões têm que estar disponível para finalizar o processo de unificação. Este tipo de informação de estado, que indica os fatos que tiveram unificação prévia em uma regra é chamada **unificação parcial**. Uma unificação parcial para uma regra é qualquer conjunto de fatos que satisfazem os padrões da regra. Então, uma regra com três padrões terá unificações parciais para o primeiro padrão, o primeiro e o segundo padrões, o primeiro, segundo e terceiro padrões, e assim sucessivamente. A unificação parcial de todos os padrões de uma regra também será uma ativação. O outro tipo de informação de estado é a **unificação padrão**, que ocorre quando um fato satisfaz

um padrão simples em qualquer regra, sem considerar as variáveis em outros padrões que possam restringir o processo de unificação.

A desvantagem do algoritmo de RETE é o uso intensivo de memória. Uma comparação simples de todos os fatos com todos os padrões não requer memória. O armazenamento do estado do sistema usando unificações de padrão e unificações parciais, entretanto, pode consumir uma quantidade considerável de memória.

Deve-se ressaltar que o referido algoritmo usa a **similaridade estrutural** nas regras. Esta similaridade refere-se ao fato de que muitas regras podem conter padrões ou grupos de padrões similares. O algoritmo de RETE tira vantagem desta característica para aumentar a eficiência, colocando os componentes comuns juntos, de maneira que eles não sejam considerados mais de uma vez.

5.2.2 Conhecimento Procedural

CLIPS também suporta o paradigma procedural para a representação do conhecimento da mesma maneira que as linguagens convencionais (PASCAL, C, etc.) ***Deffunctions*** e funções genéricas permitem que o usuário defina novos elementos executáveis no CLIPS. ***Message-handlers*** permitem que o usuário defina o comportamento dos objetos pela especificação de suas respostas às mensagens.

5.2.3 CLIPS Orientado-a-objetos

Em uma linguagem pura orientada-a-objeto, todos os elementos de programação são objetos, os quais podem apenas ser manipulados através de mensagens. No CLIPS, a definição de um objeto é muito mais complexa: números inteiros e ponto flutuante, símbolos, *strings*, campos com valores múltiplos, endereços externos, endereços de fatos e instâncias de classes definidas pelo usuário. Todos os objetos **podem ser** manipulados por mensagens, exceto as instâncias das classes definidas pelo usuário, que **devem ser**.

5.3 FuzzyCLIPS

FuzzyCLIPS é uma *Shell* para desenvolvimento de Sistemas Especialistas baseados em lógica difusa [Nrc94]. É uma versão estendida do CLIPS 6.02 (*Software* desenvolvido pelo *Johnson Space Center* - NASA, para desenvolvimento de Sistemas Especialistas) e foi implementado pelo Laboratório de Sistemas de Conhecimento do *National Research Council*, no Canadá, para a representação e manipulação de fatos e regras difusas. FuzzyCLIPS modela o raciocínio exato, *fuzzy*, e combinado, permitindo que termos difusos e normais sejam misturados livremente nas regras e fatos de um Sistema Especialista. O sistema usa dois conceitos difusos básicos: *fuzziness* e incerteza. A sua base de conhecimento está dividida em base de regras e base de fatos. As regras e fatos ficam armazenados em módulos independentes a fim de facilitar a manutenção do sistema. A figura 5.5 exibe a tela principal do FuzzyCLIPS.

Para facilitar a programação, o FuzzyCLIPS fornece algumas construções internas. As principais construções estão descritas na tabela 5.

Tabela 5 - Construções Internas do FuzzyCLIPS.

defglobal	define e inicializa as variáveis globais do sistema.
deftemplate	define as variáveis lingüísticas e seus respectivos conjuntos difusos.
deffunction	define as funções criadas pelo usuário.
defrule	define as regras que são usadas pelo sistema.
deffacts	define os fatos que inicializarão o sistema.

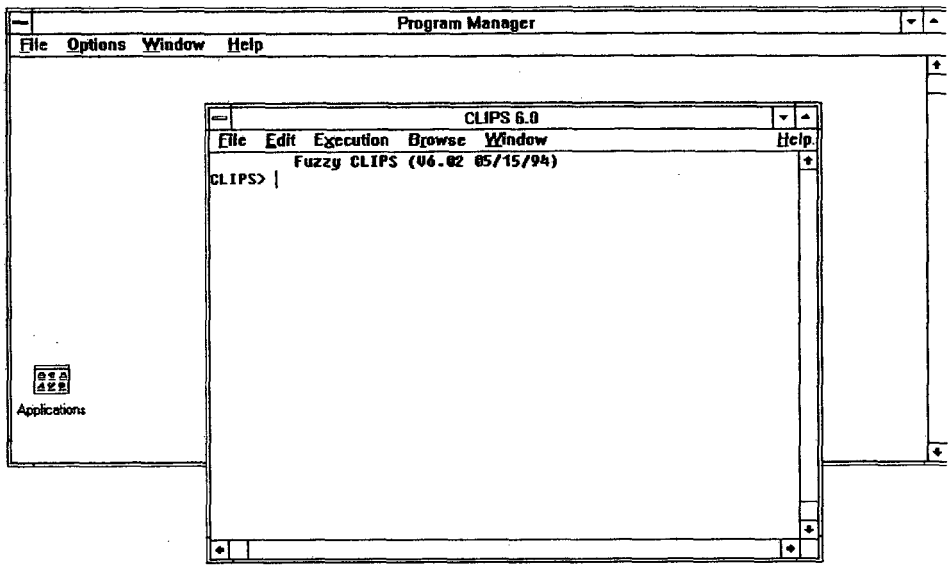

Figura 5.5 Tela Principal do FuzzyCLIPS.

As variáveis globais apresentam uma sintaxe particular que as diferencia das demais variáveis:

variáveis globais: ?*nomedavariável* demais variáveis: ?nome da variável

Tanto os fatos *crisp* quanto os difusos são representados da mesma maneira, por exemplo: (luz apagada) → fato *crisp*; (temperatura morna) → fato difuso.

Cada fato pode ser acompanhado ou não pelo seu fator de certeza (CF), como mostra o exemplo: (altura baixo) 0.6 - onde 0.6 é o fator de certeza. O valor *default* é 1.

5.3.1 As Técnicas de Inferência Usadas no FuzzyCLIPS

A avaliação da regra depende de um número de diferentes fatores, tais como se as variáveis difusas são encontradas ou não no antecedente ou consequente da regra, se uma regra contem múltiplos antecedentes ou consequentes, se um fato difuso sendo declarado, tem a mesma variável difusa que um fato difuso já existente e etc. O FuzzyCLIPS trabalha com dois tipos de regras: simples e compostas.

5.3.1.1 Regras Simples:

Considere uma regra simples da forma:

Se A então C	CF_r
A'	CF_f
C'	CF_c

onde: A = antecedente da regra;
C = consequente da regra;
CF_r= fator de certeza da regra;
CF_c= fator de certeza da conclusão.
A' = fato de união na base de fatos;
C' = atual consequente calculado;
CF_f= fator de certeza do fato;

Três tipos de regras simples são definidas: CRISP_, FUZZY_CRISP e FUZZY_FUZZY.

Se o antecedente da regra não contem um objeto difuso, então o tipo da regra é CRISP_, indiferente se ou não o consequente contem um fato difuso. Caso apenas o antecedente contenha um fato difuso, então o tipo de regra FUZZY_CRISP. Se ambos antecedente e consequente contêm fatos difusos, então o tipo de regra é FUZZY_FUZZY.

- Regra simples CRISP:

A' tem de ser igual a A, para que esta regra seja disparada. Esta é uma regra normal. Neste caso a conclusão C' é igual a C, e : $CF_c = CF_r * CF_f$

Dada uma regra:

```
(defrule exemplo1
     (declare (CF 0.7)) ; regra simples com fator de certeza 0.7
     (luz  apagada)     ; antecedente crisp
⇒
     (assert (nível_de_iluminação  escuro)) ); consequente difuso
```

e dado que o fato: (luz apagada) CF 0.8 tenha sido declarado.

Então o seguinte fato será declarado dentro de um banco de fatos devido ao disparo da regra simples crisp: (nível_de_iluminação escuro) CF 0.56, onde o fator de certeza da conclusão foi calculado como a seguir: $CF_c = 0.7 * 0.8$

- Regra simples FUZZY_CRISP

Se o tipo de regra é FUZZY_CRISP, então A' tem de ser um fato difuso com a mesma variável difusa como especificada em A, a fim de que uma união ocorra e a regra seja colocada na agenda. Quando os valores das variáveis difusas A e A' representadas pelos conjuntos difusos F_α e $F_{\alpha'}$ não forem iguais, elas devem se sobrepor. Por exemplo, os fatos difusos (temperatura alta) e (pressão alta) não se unificam porque as variáveis não são as mesmas. Entretanto, dados os fatos difusos (pressão baixa), (pressão média) e (pressão alta), claramente (pressão baixa) e (pressão

média) se sobrepõem e então se unificam, enquanto (pressão baixa) e (pressão alta) nunca se unificam.

Para uma regra deste tipo, a conclusão C' é igual a C, e $CF_c = CF_r * CF_f * S$, onde S é uma medida de similaridade entre os conjuntos difusos F_α (determinado pelo padrão difuso A) e F_α' (da unificação do fato A'). A medida de similaridade é baseada na medida de possibilidade P e na medida de necessidade N. Se a similaridade entre os conjuntos difusos associados com o padrão difuso (A) e o fato de unificação (A') é alta, o fator de certeza da conclusão é muito próximo de CF_r*CF_f se S for próximo de 1. Se os conjuntos difusos são idênticos, então S é 1 e o fator de certeza da conclusão será igual a $CF_r * CF_f$. Se a unificação é pobre, então isto é refletido em um baixo fator de certeza para a conclusão. Nota-se também que se os conjuntos difusos não se sobrepõem, então a medida de similaridade será zero. neste caso, a conclusão não seria declarada e a unificação seria considerada como tendo falhado e a regra não seria colocada na agenda.

```
(defrule exemplo2
     (declare (CF 0.7))    ; a regra tem CF = 0.7
     (fuzzy_fact  fact2)   ; antecedente difuso
⇒
     (assert (crisp_fact  fact3)))  ; consequente crisp
```

onde (fuzzy_fact fact1) CF 0.8 é o fato de união na base de fatos.

- Regra simples FUZZY_FUZZY:

Se a regra é deste tipo e o fato difuso e o padrão antecedente difuso unem-se da mesma maneira que na regra FUZZY_CRISP, então, o antecedente e o consequente de tal regra são conectados pela relação difusa: $R = F_\alpha * F_c$

onde: F_α é um conjunto difuso denotando o valor do padrão antecedente difuso.

F_C é um conjunto difuso denotando o valor do consequente difuso.

O cálculo da conclusão é baseado na regra composicional de inferência, que pode ser descrita como: $F_C' = F_\alpha' \circ R$, onde F_C' é um conjunto difuso denotando o valor do objeto difuso do consequente.

O fator de certeza da conclusão é calculado de acordo com : $CF_C = CF_r * CF_f$

```
(defrule exemplo3  ; antecedente e consequente são objetos difusos
          (temperatura quente)
⇒
          (assert (mudança pequena)))
```

5.3.1.2 Regras Compostas:

- Consequentes múltiplos:

No FuzzyCLIPS, a parte consequente da regra pode ter apenas padrões múltiplos (C1,C2,... Cn) com a conjunção AND implícita entre eles. Eles são tratados como regras múltiplas com um simples consequente. Então a seguinte regra:

SE Antecedentes então C1
SE Antecedentes então C2

SE Antecedentes então Cn

- Antecedentes múltiplos:

A partir do que foi citado acima, apenas o problema de padrões múltiplos no antecedente com uma simples afirmação no consequente necessita ser considerado. Se a afirmação consequente é um fato não difuso, um tratamento especial é necessário se a conclusão for um fato crisp. Entretanto,

se a afirmação é consequente de um fato difuso, o fator difuso é calculado usando o seguinte algoritmo básico: Se o AND lógico é usado, tem-se:

Se A1 and A2 então C	CFr
A'1	CFf1
A'2	CFf2

C'	CFc

onde A'1 e A'2 são fatos (crisp ou difuso) que unem os antecedentes A1 e A2 respectivamente. Neste caso o conjunto difuso descrevendo o valor de uma afirmação difusa na conclusão é calculado de acordo com a fórmula: $F'c = F'c1 \cap F'c2$; onde:

$\cap$ - representa a intersecção entre conjuntos difusos ($\mu_C(x) = \min(\mu_A(x),\mu_B(x))$, para $x \in U$), onde A e B são conjuntos difusos.

- F'c1 é resultado da inferência difusa para o fato A'1 e a simples regra SE A1 então C.
- F'c2 é o resultado da inferência difusa para o fato A'2 e a simples regra SE A2 então C.

O fator de certeza da conclusão é calculado de acordo com o Modelo MYCIN: CFc = min(CF'f1, CF'f2) * Cfr, onde min denota o mínimo de dois números e :

- CF'f1 é o CF de uma simples regra se A1 então C dado o fato de unificação A'1;
- CF'f2 é o CF de uma simples regra se A2 então C dado o fator de unificação A'2.

5.3.2 Limiar do fator de certeza

No FuzzyCLIPS é possível estabelecer um valor limiar para o fator de certeza tal que uma regra será disparada somente se o seu **valor de fator de certeza calculado** é maior ou igual ao valor do limiar. Esta característica pode ser útil na prevenção de um encadeamento de regras com certeza muito baixa e pouca contribuição lógica a partir do disparo, aumentando a velocidade do tempo de execução .O padrão é não ter limiar do fator de certeza, e as regras serem disparadas como usuais. O

cálculo do fator de certeza é : CFregra * min(Cf1,..., Cfn); onde CFregra é o fator de certeza para a regra e CFi são os fatores de certeza para os fatos que unificam os n padrões no lado esquerdo de uma regra.

```
(defrule abaixo_do limiar
     (declare (CF 0.5))          ; Fc da regra
     (fuzzy_fact  fato_antec.)   ; antecedente difuso
⇒
     (assert (crisp_fato c1))    ; conseq. crisp
     (assert (outro_fato_difuso c2))) ; conseq. difuso
```

Supondo que o seguinte fato foi afirmado: (fuzzy_fato fact_list_fact) CF 0.6. O fator de certeza calculado para a regra é: CF = 0.5 * 0.6 = 0.3 . A regra disparará apenas se o limiar do fator de certeza for menor ou igual que 0.3 na hora em que a regra é selecionada para disparar.

O fator de certeza para (outro_fato_difuso c2), é: CF = 0.5 * 0.6 = 0.3. Entretanto, o fator de certeza para (crisp_fato c1) é: CF = 0.5 * 0.6 * S; onde S é a medida de similaridade. Suponha que S = 0.8. Então as seguintes conclusões são alcançadas no lado direito da regra:

```
... ⇒
     (assert (crisp_fact c1));  calculado o CF=0.24
     (assert (another_fuzzy_fact c2)); calculado o CF=0.3
```

5.3.3 Defuzificação

No FuzzyCLIPS, o usuário tem a opção de escolher entre COG e MOM para defuzificar um conjunto difuso. Para isto, basta usar uma das duas funções descritas à seguir.

```
(moment-defuzzify  ?variável) ; centro de gravidade
(maximum-defuzzify ?variável) ; média dos máximos
```

Para ilustrar o processo de defuzificação, tem-se:

```
(defrule defuzificação
     ?f <-  (temperatura ?)   ; ? é usado para assegurar a
                              ;unificação do fato temperatura
⇒
     (bind ?t  (maximum-defuzzify ?f)   ;transfere o resultado da
                                        ;defuzificação da variável
                                        ;?f para a variável ?t
     (printout t "Temperatura é : " ?t crlf) ; imprime o resultado
)
```

5.3.4 A Representação dos Conjuntos Difusos no FuzzyCLIPS

Os conjuntos difusos podem ser representados no FuzzyCLIPS através de quatro formas diferentes: notação *Singleton*, e números difusos do tipo S, Z e Π.

5.3.4.1 Notação *Singleton*

O grau de pertinência $\mu_A(x)$ de x em um conjunto difuso A é um número positivo e o par $(\mu_A(x), x)$ será chamada *singleton* (sempre os pares são representados por $\mu_A(x)/x$ ou $\mu(x)/x$ para simplificar).

No FuzzyCLIPS considera-se o universo de discurso como sendo uma faixa de uma linha de números reais e não lidam com conjuntos finitos para U. Um *singleton* será representado aqui como

um par $(x_i \quad \mu(x_i))$. Um conjunto difuso A será representado como uma lista de *singletons:* <singletons> : : = $(x_1 \quad \mu_1)$ $(x_2 \quad \mu_2)$ .... $(x_n \quad \mu_n)$. Um exemplo desta representação é ilustrado a seguir:

```
(deftemplate grupo; uma declaração de variável lingüística
     0   9          ; limites do universo de discurso
     (              ; início das declarações dos termos primários
                    ; um termo primário "few" descrito na notação de
                    ;singleton

     (few (1 0) (2  0.3) (3  0.9) (4 1) (5  0.8) (6  0.5) (7 0))

     )              ; fim das declarações dos termos primários
     ( )            ; nenhum  modificador declarado
)                   ; fim do deftemplate difuso
```

5.3.4.2 Funções de representação da pertinência

Frequentemente é útil descrever uma função de pertinência usando um conjunto difuso de uma das funções S, Π ou Z. Os parâmetros destas funções podem ser escolhidos, dependendo da aplicação. Os nomes usados sugerem o formato das funções. O exemplo abaixo corresponde à representação das funções acima citadas:

```
(deftemplate Tx  "temperatura agua"
     5  65 Celsius
     ((frio     (z 10  26))
      (ok        (PI  2  36))
      (quente (s 37  60))))
```

5.4 Etapas para Construção de um Sistema Especialista Utilizando FuzzyCLIPS

Para construir um sistema especialista no FuzzyCLIPS, algumas etapas devem ser desenvolvidas:

1) Definição das Variáveis Globais

As variáveis que serão utilizadas por várias funções e módulos no sistema, bem como as variáveis manipuladas pelas funções de *interface*, devem ser definidas e inicializadas no início do programa através da construção ***defglobal***.

```
(defglobal
      ?*variavel1*       =  0
      ?*variavel2*     = " "
      .......)
```

2) Definição das Funções Criadas pelo Usuário

Todas as funções criadas pelo usuário ou engenheiro do conhecimento para *interface* com usuário, cálculos, etc., são definidas através da construção ***deffunction*** logo após a definição das variáveis globais.

```
(deffunction pergunta-simbolo (?pergunta $?valores-permitidos)
      (bind ?res (get-text-from-user ?pergunta))
      (bind ?resposta none)
      (if (eq ?res "")
            then (halt)
                  else (bind ?resposta (string-to-symbol (lowcase ?res)))
      )
      ?resposta)
```

3) Definição dos Conjuntos Difusos Utilizados pelo Sistema

Todas as variáveis lingüísticas utilizadas no sistema têm seus conjuntos difusos determinados através da construção ***deftemplate***. Aqui também define-se o universo de discurso das variáveis lingüísticas.

```
(deftemplate VL
      0 10 ud
(     (conjdifuso1 (0 0) (1 1) (2 0))
      (conjdifuso2 (1 0) (2 1) (3 0))
      (conjdifuso3 (2 0) (3 1) (4 0))
      (conjdifuso4 (3 0) (4 1) (5 0))
      ...) )
```

4) Regra de Startup

Este passo consiste em determinar a regra que iniciará a execução do programa. Esta regra é diferenciada das outras pela ausência de antecedente. Dentro do consequente desta regra deve constar os módulos onde se encontram as regras e as chamadas das funções a serem usadas durante a execução do sistema.

```
(defrule startup
=>
      (load "teste1.clp")
      (pergunta-simbolo)
      ....)
```

5) Definição das Regras

As regras a serem utilizadas pelo sistema devem ser definidas através da construção ***defrule.*** Estas regras podem ser definidas no corpo do programa principal (se forem em pequeno número), mas geralmente são definidas em módulos separados, os quais são carregados na regra de *startup*.

```
(defrule exemplo
      (VL conjdifuso1)
=>
      (printout t "Bem Vindo ao FuzzyCLIPS))
```

Estas são as etapas básicas, que devem constar em todos os programas em FuzzyCLIPS e estão sintetizadas na tabela 5. Outras etapas podem ser necessárias, dependendo da necessidade da aplicação.

5.5 wxCLIPS

wxCLIPS é uma ferramenta desenvolvida pelo AIAI - *Artificial Intelligence Applications Institute*, Universidade de Edinburgh, Reino Unido, em 1994 [Sma94]. O desenvolvimento de tal ferramenta foi motivado pela necessidade de suprir uma deficiência dos programas feitos em CLIPS ou FuzzyCLIPS - a *interface* do usuário.

wxCLIPS pode ser considerado em dois aspectos: (i) uma biblioteca de funções CLIPS/FuzzyCLIPS, para acessar as facilidades do wxWindows; (ii) um ambiente de desenvolvimento de aplicações wxWindows, usando as funções CLIPS/FuzzyCLIPS. A biblioteca pode ser usada por qualquer programa C++; wxCLIPS é então, uma simples *interface* de desenvolvimento *for Windows*, que usa a biblioteca de funções CLIPS/FuzzyCLIPS.

5.5.1 Ambiente de Desenvolvimento wxCLIPS

wxCLIPS possui um ambiente de desenvolvimento básico, que consiste em uma janela com *menus*, uma janela para entrada de dados e uma janela para os textos de saída. Durante a compilação dos programas, as mensagens de erro são escritas na janela de saída.

Usando o wxCLIPS, pode-se criar *frames*, cada um com seus próprios *menus*. Dentro de um *frame* pode-se criar uma ou mais subjanelas. Estas subjanelas podem ser *panels*, *canvases* e subjanelas de texto.

Panels são usados para conter os *panel items*, tais como botões, itens para entrada de texto, box de listas, etc. O box de diálogo é uma forma especial de *panel*, que contem seu próprio *frame*, sendo assim, ao invés de criar um *frame* e um *panel*, basta criar um box de diálogo e inserir os *panel items* necessários. *Canvases* são usados para desenhar figuras com qualquer formato.

As subjanelas de texto são usadas para exibir arquivos texto ou editar os programas.

Não há necessidade de criar um box de diálogo ou manipular todos os botões, e outros eventos manualmente. Há um certo número de funções que desempenham todas estas tarefas, tais como: **get-text-from-user, message-box, get-choice, file-selector**, etc. Tais funções bloqueiam o fluxo de execução do programa no ponto onde foram chamadas, até que o usuário responda às solicitações. A figura 5.6 exibe algumas das opções de *interface* que o wxCLIPS fornece.

Há uma única função obrigatória a todos os programas que são executados no wxCLIPS: **app-on-init**. Se uma aplicação define uma função com este nome, a *interface* wxCLIPS do usuário

pode inicializar a aplicação e estabelecer todas as variáveis de ambiente, arquivos e funções que serão usadas no decorrer da execução.

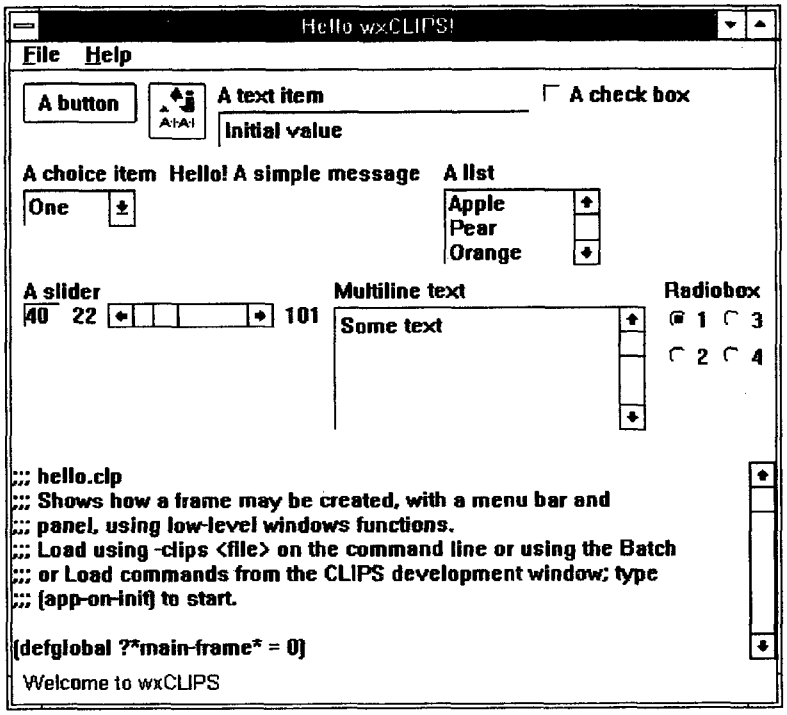

Figura 5.6 Exemplos de *Interfaces* do wxCLIPS.

CAPÍTULO VI

Aplicação de um Sistema Especialista Difuso no Processo de Análise Química Qualitativa de Amostras de Minerais

6.1 Introdução

Uma das questões mais importantes, dentro do universo de aplicações dos sistemas especialistas, é a modelagem do conhecimento de maneira à se aproximar do raciocínio humano.

A lógica difusa surge como uma solução para este tipo de problema, uma vez que permite o raciocínio aproximado. Desta forma, os sistemas especialistas difusos levam a resultados bem mais satisfatórios do que os sistemas especialistas tradicionais.

Nos capítulos anteriores foram analisadas técnicas que têm sido aplicadas, com sucesso, no desenvolvimento de sistemas especialistas. A união de sistemas especialistas e conjuntos difusos constitui-se em uma forma poderosa para modelar o conhecimento do especialista.

Neste capítulo, foi desenvolvida uma aplicação que possibilita a compreensão de como essas ferramentas podem ser aplicadas a problemas reais. O problema escolhido é o processo de análise química qualitativa de amostra de minerais. A solução é o desenvolvimento de um sistema especialista difuso que possa ser utilizado nesta análise.

6.2 Descrição do Universo de Estudo

Para demonstrar a aplicabilidade da lógica difusa em sistemas especialistas, bem como o uso da ferramenta FuzzyCLIPS, escolheu-se o processo de "Análise Química Qualitativa" para amostras de minerais.

Tal universo de estudo foi escolhido devido ao tipo de incerteza que o cerca; à sua necessidade de conhecimento especialista; e por envolver ponderações e inferências cognitivas peculiares ao ser humano.

O objetivo da análise química qualitativa é determinar a composição qualitativa das substâncias. Este tipo de análise é de enorme importância científica e prática, sendo um dos mais importantes métodos de investigação de substâncias. A química analítica está presente em vários campos da ciência, tais como mineralogia, geologia, fisiologia, bem como medicina, agricultura, etc., ou seja, ela é indispensável no controle químico da produção industrial, investigação de solos, fertilizantes, produtos agrícolas, minerais industrializáveis, indústria do petróleo, indústria metalúrgica, etc [For87].

Muitas vezes é necessária a análise mineralógica da amostra para auxiliar a análise química qualitativa, devido à grande semelhança entre os vários tipos de minerais. A análise mineralógica consiste em ensaiar dureza, densidade, brilho, fusibilidade, cor e sistema de cristalização da amostra [Anexo I].

6.3 O Problema

A região do Vale do Rio Doce é uma região pegmatítica onde ocorre uma variedade enorme de minerais considerados pedras semi - preciosas, além de minerais industriais como o feldspato, o quartzo, a dolomita, a columbita, o berilo, etc. Da mesma forma, o Vale do Mucuri e do Jequitinhonha têm formação pegmatítica. Em decorrência dessa formação geológica, a mineração de lavras em busca de gemas para exportação é bastante intensa. Quando ocorre alguma amostra diferente, os garimpeiros buscam na Universidade Vale do Rio Doce - UNIVALE informações para que possam orientar adequadamente suas atividades.

As informações buscadas dizem respeito a que tipo de mineral correspondem as amostras encontradas, o que demanda além de um exame mineralógico preliminar, uma análise química da amostra para que se possa definir o mineral representado pela mesma.

As amostras chegam ao INPRO (Instituto de Projeto) trazidas pelo interessado, que após explicar o que deseja, é encaminhado ao Laboratório de Química Analítica do CECET - Centro de Ciências Exatas e Tecnológicas, onde a amostra é identificada, recebendo uma etiqueta discriminando o tipo de análise a que se deve submeter.

Em alguns casos, faz-se um exame mineralógico auxiliar para orientação do analista, uma vez que a maioria dos contratantes dos serviços de análise são pequenos mineradores da região. Na maioria das vezes, a região de procedência da amostra é um dos principais indicadores de que tipo de mineral se trata. No entanto, por ser a região pegmatítica, onde há um sem número de minerais que se parecem macroscopicamente, e na maioria das vezes com propriedades físicas similares, a incerteza domina a análise.

Utilizando as informações procedentes da análise mineralógica, o analista passa aos testes de

solubilidade para que possa praticar os ensaios químicos propriamente ditos. Obtida a solução inicial e após eliminar o silício comum a todo e qualquer mineral, passa-se a uma série de testes químicos para a separação de grupos analíticos e dentro de cada grupo a identificação de um determinado elemento químico. Cabe ressaltar que dentro de um mesmo grupo analítico, os elementos químicos poderão ter três, quatro ou mais reações comuns, sendo a diferenciação feita muitas vezes por uma única reação química específica [For87].

O problema da análise de uma amostra de mineral começa na fase da análise mineralógica, já que os resultados dos ensaios podem ser os mesmos para vários minerais, como por exemplo, se forem comparados os ensaios mineralógicos da hornblenda e da tremolita, chega-se aos resultados da tabela 6.

Tabela 6 - Comparação da Análise Mineralógica da Hornblenda e da Tremolita.

	Hornblenda	Tremolita
Cristalização	monoclínico	monoclínico
Densidade	2.9 - 3.4	2.9 - 3.2
Cor	Branca	Branca
Brilho	Vítreo	Vítreo
Dureza	5 - 6	5 - 6
Fusibilidade	Almandita - Actinolita	Actinolita

A diferença mais sensível entre as duas amostras só é verificada após a análise química, já que as duas diferem apenas na composição química:

Hornblenda: Ca, Al, Mg, Fe^{+2}, Fe^{+3}; *Tremolita:* $CaMg_3(SiO_3)_4$.

Como se pode verificar, a primeira etapa da análise (análise mineralógica) possue resultados bastante incertos, variando de observador para observador e das condições de observação e da amostra.

Já na etapa da análise química qualitativa, os problemas começam com o teste de solubilidade, pois o químico deve saber qual o solvente específico para cada tipo de mineral. São inúmeros os tipos de minerais para cada solvente. Se o químico não tiver certa prática, provavelmente, muito tempo será perdido consultando as fontes necessárias para descobrir o solvente adequado.

Logo após a solubilização, o problema se agrava, uma vez que a solução obtida deverá ser submetida a uma série de tratamentos químicos que irão definir os diversos cátions e ânions e a seguir, identificar cada um deles. Sendo assim, o químico deve ter um vasto conhecimento para escolher qual a próxima etapa a seguir.

Diante destes problemas, fica provado que uma pessoa com pouca experiência, como químicos ou técnicos recém formados, ou até mesmo químicos que tenham especialização em outras áreas que não seja a química analítica, gastarão bastante tempo analisando e pesquisando cada passo e cada resultado, ficando a rapidez e a confiabilidade do processo vinculados a um especialista.

6.4 Solução Proposta

A partir de todos os problemas que são verificados diariamente em um processo de análise química qualitativa de uma amostra de mineral, a necessidade de um Sistema Especialista para auxiliar toda a tarefa é bastante visível.

Foi implementado então, um sistema especialista capaz de acompanhar todo o processo (análises mineralógica e química) sem deixar de lado toda a incerteza que cerca esta análise, já que principalmente no que diz respeito à análise mineralógica, o resultado da observação depende muito da perícia do observador, variando bastante de analista para analista, principalmente no que diz

respeito à cor, dureza e brilho.

Para tratar este problema de incerteza, o Sistema Especialista foi estruturado com base na lógica difusa.

6.4.1 Aquisição do Conhecimento

O processo de aquisição do conhecimento para o sistema descrito neste trabalho iniciou-se em fevereiro de 1992. Durante seis meses o especialista foi observado em seu local de trabalho, para que o engenehiro do conhecimento pudesse compreender de forma correta como é desenvolvida análise química qualitativa de amostras de minerais.

Terminada a fase de observação, iniciou-se a etapa de entrevistas, a fim de elucidar alguns pontos críticos da análise, que não ficaram bem explicitados durante a primeira fase.

A seguir, iniciou-se a estruturação das regras que compõem a base de conhecimento. Tais regras foram estruturadas, inicialmente, com base na lógica clássica. Tal etapa finalizou-se em 1993.

Em março de 1994 iniciou-se a conversão das regras para a lógica difusa. Os conjuntos difusos foram estabelecidos de acordo com a perícia do especialista. Encerrada esta etapa, as regras foram revisadas pelo especialista, e as incoerências foram eliminadas.

Verificou-se, durante esta fase de revisão, que ficou mais fácil para o especialista interagir com o engenheiro do conhecimento à partir do momento em que foi feito uso de variáveis

linguísticas, ou seja, à partir do uso da lógica difusa, uma vez que desta maneira, ele pode se expressar de maneira mais próxima do seu raciocínio.

6.4.2 Implementação do Sistema

A implementação do Sistema Especialista para Análise Química Qualitativa de amostras de minerais foi dividida em três etapas: (i) estudo das características a serem consideradas e determinação dos conjuntos difusos necessários; (ii) estruturação das regras; (iii) desenvolvimento da *interface* do usuário.

6.4.2.1 Determinação do Conjuntos Difusos

Após um levantamento das características manipuladas durante o processo de análise mineralógica, conclui-se que algumas delas deveriam ser representadas sob a forma de conjuntos difusos, devido à natureza imprecisa das mesmas. A cor da amostra não é tratada através de conjuntos difusos porque a sua importância na análise mineralógica não é relevante.

As características escolhidas para serem representadas através de conjuntos difusos foram: dureza, densidade, índice de refração e fusibilidade [AnexoI]. O brilho e a cor foram considerados como variáveis *crisp*. Os conjuntos difusos correspondentes a cada uma destas variáveis são demonstrados à seguir.

```
(deftemplate DUREZA
      0 10 ud
  (   (dr_ndet           (0 1) (0.1 0))
      (talco       (0 0) (1 1) (2 0))
      (gipso             (1 0) (2 1) (3 0))
      (calcita           (2 0) (3 1) (4 0))
      (fluorita          (3 0) (4 1) (5 0))
      (apatita           (4 0) (5 1) (6 0))
      (ortoclasse        (5 0) (6 1) (7 0))
      (quartzo           (6 0) (7 1) (8 0))
      (topazio           (7 0) (8 1) (9 0))
      (coridom           (8 0) (9 1) (10 0))
      (diamante          (9 0) (10 1))
  ) )

(deftemplate FUSIBILIDADE
      0 7 uf
     ((f_ndet        (0 1) (0.1 0))
      (estibinita        (0 0) (1 1) (2 0))
      (natrolite         (1 0) (2 1) (3 0))
      (garnet_lima       (2 0) (3 1) (4 0))
      (hornblende        (3 0) (4 1) (5 0))
      (ortoclasse        (4 0) (5 1) (6 0))
      (bronzite          (5 0) (6 1) (7 0))
      (infundivel        (6 0) (7 1))
     )  )

(deftemplate DENSIDADE
      0 25
  ((dn_ndet        (0 1) (0.1 0))
   (db       (0 0) (6 1) (12 0))
   (dm       (6 0) (16 1) (20 0))
   (da       (10 0) (20 1) (25 1))
  ) )

(deftemplate REFRACAO
   0 4
  (   (r_ndet       (0 1) (0.1 0))
      (refb         (0 0) (1 1) (2 0))
      (ref          (1 0) (2 1) (3 0))
      (refa         (2 0) (3 1) (4 1))
  ) )
```

6.4.2.2 Estruturação das Regras

Após a determinação dos conjuntos difusos, foram estruturados dois tipos de regras: (i) regras relacionadas a análise mineralógica; (ii) regras relacionadas a análise química, num total de quinhentos e quarenta regras.

As regras relacionadas a análise mineralógica foram estruturadas de maneira à colocar em uma mesma regra, os minerais que possuíam características mineralógicas bastante semelhantes. Como exemplo, tem-se o caso ilustrado a seguir.

```
 (defrule R15-Altaita-Amalgama-Canfieldita-Claustalita-Cobre-
Petzite-Tiemannite

   (DUREZA calcita)
   (DENSIDADE dm)
   (REFRACAO r_ndet)
   (BR "m")
   (FUSIBILIDADE f_ndet)
   (CRISTAL "c")
=>
   (assert (composicao Altaita Pb))
   (assert (mineral Altaita))
   (assert (solub Altaita soluvel-hno3))
   (assert (composicao Amalgama Ag))
   (assert (composicao Amalgama Hg))
   (assert (mineral Amalgama))
   (assert (solub Amalgama soluvel-hno3))
   (assert (composicao Canfieldita Ag))
   (assert (composicao Canfieldita Sn))
   (assert (mineral Canfieldita))
   (assert (solub Canfieldita soluvel-hno3))
   (assert (composicao Claustalita Pb))
   (assert (mineral Claustalita))
   (assert (solub Claustalita soluvel-h2so4))
   (assert (composicao Cobre Cu))
   (assert (mineral Cobre))
```

```
(assert (solub Cobre soluvel-hno3))
(assert (composicao Petzite Ag))
(assert (composicao Petzite Au))
(assert (mineral Petzite))
(assert (solub Petzite soluvel-hcl))
(assert (composicao Tiemannite Hg))
(assert (mineral Tiemannite))
(assert (solub Tiemannite soluvel-agua-regia)))
```

Os sete elementos citados na regra R15 (Altaita-Amalgama-Canfieldita-Claustalita-Cobre-Petzite-Tiemannite) possuem características bastante semelhantes, levando à composição dos mesmos conjuntos difusos. Esta capacidade de agrupar os elementos semelhantes na mesma regra, torna os sistemas baseados na lógica difusa bem mais rápidos e mais compactos do que os sistemas tradicionais.

As regras relativas à análise química propriamente dita, são baseadas no estudo da cor dos precipitados e têm como resultado final, a determinação dos cátions que compõem a amostra. Um exemplo destas regras é ilustrado à seguir através de um pequeno trecho da regra que lida com a determinação do cátion de mercúrio (Hg^{++}).

```
(defrule Marcha4
        (declare (salience -400))
        (grupo ?m grupo2)
=>
        (printout t "ANALISE DO GRUPO II" crlf crlf)
        (printout t "Este grupo e' subdividido em II-A e II-B"          crlf)
        (printout t "II-A: possiveis presencas de: Cu++, Pb++,
Bi+++, Cd++, Hg++" crlf)
        (printout t "II-B: possiveis presencas de: Sn++, Sn4+,          As3+,
As5+, Sb3+, Sb5+" crlf crlf)
        (printout t "Para analisar este grupo, deve-se tratar o        ppt  do
Passo 2 a etapa inicial, " crlf)
        (printout t "com sulfeto de amonio e filtrar. Havera
formacao de um ppt preto e um filtrado." crlf)
        (printout t "No residuo preto inicia-se o estudo do Grupo       II-A"
crlf)
```

```
        (printout t "No filtrado inicia-se o estudo do Grupo II-B"          crlf
crlf)
        (printout t "ESTUDO DO GRUPO II-A" crlf)
        (printout t "PASSO 1" crlf)
        (printout t "[1] Tratar o residuo preto por 10 ml de HNO3
[1:2]" crlf)
        (printout t "[3] Ferver e filtrar. Observar a cor do ppt"           crlf
crlf)
            (bind ?color (cor-ppt "Qual a cor observada no ppt?"))
            (if (eq ?color preto)
                then
                 (printout t "Possivel presenca de Hg++" crlf
         crlf)
                 (printout t "Efetue um dos seguintes ensaios para
         confirmar a presenca de Hg++" crlf)
                 (prepara-lista "Ensaios para confirmacao de
         Hg++")
                 (preenche-lista-g2-a-hg)
                 (bind ?ensaio (executa-lista))
                 (if (eq ?ensaio "reagir-com-gas-sulfidrico")
                    then
                     (bind ?color (cor-ppt "Qual a cor observada
        no ppt?"))
                     (if (eq ?color preto)
                         then
                          (printout t "Confirmada a presenca de
              Hg++" crlf)
                          (assert (cations ?m hg++))
                         else
                          (printout t "Ausencia de Hg++" crlf)
                     )
                 )
                 (if (eq ?ensaio "reagir-com-cloreto-estanoso")
                    then
                     (bind ?color (cor-ppt "Qual a cor observada
no ppt?"))
                     (if (or (eq ?color branco) (eq ?color cinza-
        escuro))
                         then
                          (printout t "Confirmada a presenca de
              Hg++" crlf)
                          (assert (cations ?m hg++))
```

```
        else
         (printout t "Ausencia de Hg++" crlf)
) )
```

A implementação deste sistema utilizando a *shell* FuzzyCLIPS foi facilitada devido ao ambiente de programação fornecido [Anexo II]. Um problema porém, foi detectado. O editor desta ferramenta tem capacidade limitada. Somente metade das regras utilizadas pelo sistema (250), poderam ser editadas diretamente neste editor, devido a falta de memória. Desta forma, optou-se por editar as regras em no editor de texto do DOS: EDIT.

6.4.2.3 *Interface* do Usuário

Outra limitação encontrada no FuzzyCLIPS propriamente dito, é a ausência de uma *interface* apropriada para o usuário. Sendo assim, necessitou-se utilizar o FuzzyCLIPS compilado junto com outra ferramenta, o **wxCLIPS** [Sma94], desenvolvida justamente com o objetivo de eliminar esta dificuldade.

Uma vez compilado juntamente com o wxCLIPS, o FuzzyCLIPS pode utilizar as funções pré-definidas por esta ferramenta, podendo criar então, inúmeros tipos de *interface*. As figuras 6.1 e 6.2 ilustram algumas das *interfaces* utilizadas neste sistema.

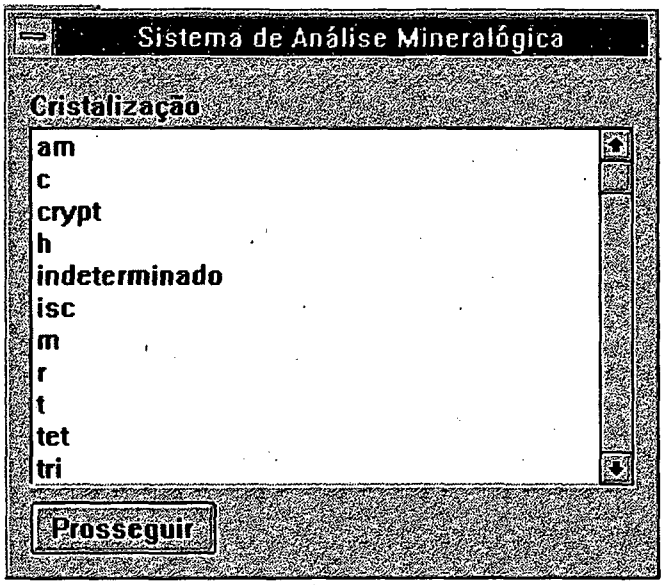

Figura 6.1 *Interface* do Usuário - Lista de Opções para o Sistema de Cristalização.

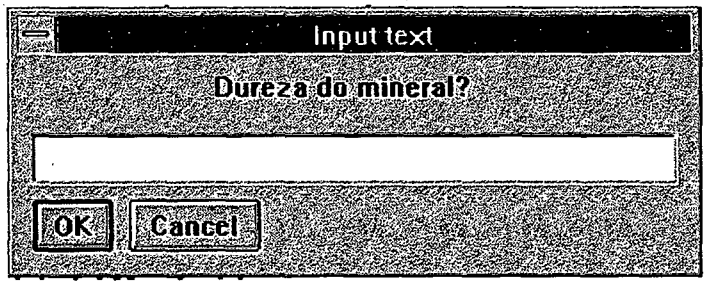

Figura 6.2 *Interface* do Usuário - Box de Diálogo para Entrada dos Valores da Variável Dureza.

6.5 Análise de Desempenho do Sistema

Uma vez concluída a implementação do sistema, inicia-se a fase de testes e análise de desempenho. O sistema é testado para determinar se as regras e objetos estõa encadeados apropriadamente e, se de fato, todas as regras presentes na base de conhecimento são necessárias. Concluídos os testes, analisa-se o desempenho do sistema. Esta análise é feita com base em uma "matriz Confusa" [Mau91].

No campo da psicologia, uma "Matriz Confusa" é construída para determinar a relação entre as respostas corretas e aquelas respostas que chegaram próximo ao acerto. No caso dos sistemas especialistas, determina-se a relação entre as respostas do especialista e as do sistema. Em uma

"Matriz Confusa", cada entrada representa a porcentagem de vezes em que o sistema difere do especialista - o que siginifica a porcentagem de vezes que o sistema especialista ficou confuso. A média dessas porcentagens de erro fornece um limiar de erro através do qual verifica-se o desempenho geral do sistema, considerando-o eficiente ou não.

A fim de avaliar o desempenho do sistema especialista implementado neste trabalho, fez-se uma bateria de testes, através da qual concluíu-se que o sistema é eficiente. Para ilustrar alguns dos resultados obtidos, tem-se o exemplo da "Matriz Confusa" referente aos minerais Hornblenda e Tremolita na tabela 7.

Tabela 7 - Matriz Confusa para Hornblenda e Tremolita

Sistema Especialista / Especialista	Hornblenda	Tremolita	
Hornblenda	n/a	37%	Acerto: 63%
Tremolita	29%	n/a	

Acerto: 71%

Após realizar esta análise, foi feita uma avaliação em conjunto com o especialista, a fim de verificar quais os prováveis causadores destes equívocos. Concluiu-se que tais problemas ocorreram devido às fronteiras estabelecidas aos conjuntos difusos. Portanto, para melhorar a performance deste sistema deve-se fazer um ajuste destas fronteiras.

6.6 FuzzyCLIPS x CLIPS

A fim de demonstrar as vantagens do uso de um sistema especialista difuso em relação aos sistemas especialistas tradicionais, uma comparação foi feita envolvendo o FuzzyCLIPS e o CLIPS, comparações estas que são descritas a seguir.

No sistema *crisp* - CLIPS, as regras foram escritas de maneira a descrever todas os possíveis valores que uma dada característica pode assumir, ou seja, se uma característica pode assumir os valores de 1.65 a 1.89, foi necessário determinar todos os valores (com duas casas decimais) que cobriam estes valores. Já no FuzzyCLIPS, isto não ocorreu, uma vez que o uso dos conjuntos difusos permite que estes intervalos já estejam mapeados, necessitando somente informar o conjunto difuso correspondente na regra. Esta característica economiza várias linhas de código, e com isto o sistema fica mais compacto e consequentemente mais rápido. Como consequência disso, a base de regras do FuzzyCLIPS é menor em relação ao CLIPS (uma diferença de cerca de cem regras).

A interação com o usuário também é facilitada no FuzzyCLIPS, uma vez que o usuário não necessita ser altamente preciso, podendo apresentar valores aproximados; o que não ocorre no CLIPS.

CAPÍTULO VII

Conclusões e Recomendações

7.1 Conclusões

Os progressos obtidos recentemente a partir da unificação da teoria dos conjuntos difusos e sistemas especialistas, têm contribuído para o aumento do potencial de aplicação dos computadores.

Os conjuntos difusos contribuem de uma forma significativa para que a modelagem da base de conhecimento dos sistemas especialistas se torne mais próxima do raciocínio humano. Isto faz com que os resultados obtidos nestes sistemas sejam mais satisfatórios.

O desenvolvimento deste trabalho abordou o estudo sobre sistemas especialistas difusos. Como exemplo de aplicação, desenvolveu-se um sistema especialista difuso para análise química qualitativa de amostras de minerais [Fer96]. O sistema desenvolvido apresenta uma modelagem difusa da base de conhecimento. Tal modelagem permite uma diminuição do número de regras da base de conhecimento, consequentemente, diminuindo o tempo de busca gasto pela máquina de inferência. Para o desenvolvimento do referido sistema, foi utilizada a *shell* FuzzyCLIPS (*Fuzzy C Language Integrated Production Systems*)

A performance do sistema provou que a aplicabilidade de sistemas especialistas difusos é válida, uma vez que sua performance mostrou-se satisfatória, se comparada a um sistema tradicional, em termos computacionais; e seus resultados comprovaram na prática, que é possível realizar a análise química qualitativa de amostras de minerais, independente da perícia do usuário.

7.2. Recomendações

Com relação ao problema proposto, recomenda-se para trabalhos futuros, a ampliação do universo a ser analisado, estendendo a análise ao nível quantitativo.

Quanto à aquisição dos dados, recomenda-se o uso de técnicas de reconhecimento de padrões, processamento de imagens [Fer95]; a utilização de redes neurais; o emprego da lógica difusa, bem como o uso do processamento paralelo.

A idéia central é a de permitir a incorporação de novas informações relevantes, que técnicas tradicionais não conseguem capturar. Através dessas novas informações, espera-se, aumento de confiabilidade dos resultados, ampliação e incorporação de novos conhecimentos a respeito do processo de análise, reduções substanciais de custo também são esperadas (com aumento de efetividade, abrangência, redução de tempo, etc.).

REFERÊNCIAS BIBLIOGRÁFICAS

[Bas94] Bastos, Rogério Cid, "Avaliação de Desempenho de Sistemas Educacionais: Uma Abordagem Utilizando Conjuntos Difusos"., Universidade Federal de Santa Catarina, Florianópolis, Brasil, 1994

[Bez93] Bezdek, James C., "Fuzzy models - what are they, and why?", IEEE - Transactions on Fuzzy Systems, 1:1, pp 1-6, USA, 1993.

[Bro85] Brownston, L., "Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming", Addison-Wesley, 1985.

[Cox94] Cox, E., "The Fuzzy Systems Handbook: a practitioner's guide to builduing, using, and maintaining fuzzy systems", Academic Press, Inc, Cambridge, USA, 1994.

[Dep90] Department of Trade and Industry - "Guidelines for the introduction of expert systems technology", HMSO, England, 1990.

[Fer95] Fernandes, Anita M. da Rocha, Bastos, Rogério Cid, "A Fuzzy Expert System to Qualitative Chemical Mineral Analyses Using Color Recognition", In: *Proceedings of ISUMA - NAFPIS'95* - Maryland, USA, 1995.

[Fer96] Fernandes, Anita M. da Rocha, Bastos, Rogério Cid, "A Fuzzy Expert System to Qualitative Chemical Mineral Analyses", In: *Proceedings of The Third World Congrees on Expert Systems* , - Seoul, Korea, Korea, 1996.

[For85] Forgy, C.L., "Rete: A Fast Algorithm for the Many Pattern? Many Object Pattern Match Problem," Artificial Intelligence, 19, pp 17-37, 1985.

[For87] Ford, W.E., Dana's Text Book of Mineralogy, 4th Edition, 1932. New York: John Willey and Sons, 24, 1987.

[Gia89] Gairratano, J., Riley, G., "Expert Systems - Principles and Programming", PWS - KENT Publishing Company, Boston, USA, 1989.

[Gon86] Gonçalves, C.A., "Aquisição e Representação do Conhecimento para Sistemas Especialistas", Faculdade de Economia e Administração da Universidade de São Paulo, São Paulo, 1986.

[Har85] Hart, A., "Knowledge Elicitation: Issues and Methods", In: *Computer Aided Design*", volume 17, número 9, Novembro de 1985.

[Jac90] Jackson, P., "Introduction to Expert Systems", MA: Addison Weley, USA, 1990.

[Kli92] Klir, G.J., Folger, T.A., "Fuzzy Sets, uncertainty, and information", McGrawHill, Inc, USA, 1992.

[Kos92] Kosko, B., "Neural Networks and Fuzzy Systems: a dynamical system approach to machine intelligence", Prentice Hall , New Jersey, USA, 1992.

[Lbj93] Lyndon B. Johnson Space Center - Software Technology Branch - "CLIPS Reference Manual - Vol I - Basic Programming Guide", USA, 1993.

[Lie94] Liebowitz, J., "Worldwide Expert Systems Activities and Trends", Cognizant Communication Offices, USA, 1994.

[Mau91] Maus, R., Keyes, J., "Handbook of Expert Systems in Manufacturing", McGrawHill, Inc, USA, 1991.

[Nrc94] National Research Council of Canada - "FuzzyCLIPS User's Guide", Canada,

1994.

[Ric91] Rich, E., Knight, K., "Artificial Intelligence", McGrawHill, Inc., New York, USA, 1991.

[Rob89] Roberts, D.W., "Analysis of Forest Sucession with Fuzzy Graph Theory", Ecological Modeling, 45:261-274, USA, 1989.

[San90] Santos, E. R., Becker, J.L., "A Questão do Conhecimento de Sistemas e sua Aquisição", Série Documentos para Estudo, PPGA/UFRGS, Fevereiro, 1990.

[Sho86] Shortliffe, E.H., "Computer - based medical consultation: MYCIN", American Elsevier, USA, 1986.

[Sma94] Smart, Julian, "User Manual for wxCLIPS 1.30", Artificial Intelligence Apllications Institute - University of Edinburg - United Kingdom, 1994.

[Taz87] Tazaki, E., "Fuzzy Expert Systems", Mathematical Science, 284, USA, 1987.

[Ter95] Terano, T., Asai, K., Sugeno, M., "Applied Fuzzy Systems", AP Professional, Japan, 1995.

[Tur91] Turksen, I,B., "Measurement of Membership Functions and their Acquisitions", *Fuzzy Sets and Systems*, USA, 1991.

[Tur92] Turksen, I,B., Wilson, I.A., "A Fuzy Set Model for Market Share and Preference Prediction", *European Journal of Operational Research*, England, 1992.

[Tur93] Turksen, I.B., Tian, Y., "Combination of Rules of their Consequences in Fuzzy Expert Systems", *Fuzzy Sets and Systems*, 58- 3 - 40, USA, 1993.

[Wat86] Waterman, D. A., "A guide of Expert Systems", Addison - Wesley Publishing Company, Inc., USA, 1986.

[Zim92] Zimmermann, H.J., "Fuzzy Sets Theory and Its Applications", Klewer - Nighoff, Boston, USA, 1992.

[Zur94] Zurada, M. Jacek, Marks II, R. J., Robenson, Charles J., "Computational Intelligence Imitating Life", IEEE Press Marketing, USA, 1994.

ANEXO I

Análise Química Qualitativa de Amostras de Minerais

1. Introdução

A análise química qualitativa de amostras de minerais tem por objetivo determinar a composição qualitativa das substâncias. Muitas vezes, a análise química necessita ser antecipada pela análise mineralógica, devido à grande semelhança entre os vários tipos de minerais. A seguir, os dois processos serão descritos [For87].

2. Análise mineralógica

- Dureza:

A dureza da amostra é determinada com base em uma "Tabela de Dureza", onde o ensaio fornece um número relativo ao grau de dureza da amostra, que corresponde a um dos rótulos exibidos na tabela 7:

Tabela 7 - Escala de Dureza.

1) talco	2) gipso	3) calcita	4) fluorita
5) apatita	6)ortoclásio	7) quartzo	8) topázio
9) coridom	10) diamante		

1) talco - corresponde à amostra que é muito facilmente riscada pela unha, e dá a sensação de untuosidade ao tato;

2) gipso - corresponde à amostra que é facilmente riscada pela unha;

3) calcita - corresponde à amostra que é riscada por um estilete de bronze ou moeda de cobre;

4) fluorita - corresponde à amostra que é facilmente riscada por uma faca;

5) apatita - corresponde à amostra que é riscada com dificuldade por uma faca;

6) ortoclásio - corresponde à amostra que é facilmente riscada por um arame;

7) quartzo - correponde à amostra que é um pouco marcada por um arame, mas irá riscar um vidro de janela.

Todas as amostras que forem mais duras que o quartzo irão riscar um vidro de janela.

Para classificar as amostras nos graus 8, 9 e 10, o analista deverá compará-las com os padrões de topázio, corindom e diamante que devem existir no laboratório, uma vez que a diferença entre eles está no seguinte fato:

O diamante não é riscado por nenhum objeto, porém risca todos. O corindom risca o topázio e os demais acima dele na escala de dureza. O topázio risca o quartzo e os demais acima dele na escala de dureza.

- Sistema de Cristalização:

É o sistema geométrico de agrupamento dos cristais. Classifica-se segundo a tabela 8:

Tabela 8 - Sistemas Geométricos.

1) C isométrico;	2) Tet tetragonal;	3) H hexagonal;
4) R ortorômbico;	5) M monoclínico;	6) Tri triclínico;
7) Am amorfo;	8) Cript criptocristalino;	9) Iso isotrópico;
10) ? duvidoso.		

O teste para determinar o sistema crsitalino é a observação microscópica da amostra, a identificação da forma dos cristais e o tipo de fratura.

- Fusibilidade:

A fusibilidade da amostra é analisada com base em uma "Tabela de Fusibilidade", que recebe após o ensaio um número relativo ao grau de fusibilidade da amostra, que corresponde a um dos seguintes rótulos exibidos na tabela 9:

Tabela 9: Escala de Fusibilidade.

1) Estibinita;	2) Calcopirita;	3) Almandita;	4) Actinolita;
5) Ortoclásio;	6) Bronzita;	7) Fundível	

1) Estibinita - corresponde à amostra que funde facilmente à chama da vela;

2) Calcopirita - corresponde à amostra que funde à chama luminosa, mas com dificuldade em um tubo fechado;

3) Almandita - corresponde à amostra que funde facilmente ao dardo da chama;

4) Actinolita - corresponde à amostra que tem as arestas facilmente arredondadas pelo dardo da chama;

5) Ortoclásio - corresponde à amostra que tem as arestas fundidas com dificuldade pelo dardo da chama;

6) Bronzita - corresponde à amostra na qual somente os fragmentos mais finos são arredondados pelo dardo da chama.

- Brilho:

É observado e classificado segundo um dos rótulos exibidos na tabela 10:

Tabela 10 - Escala de Brilho.

1) metálico;	2) vítreo;	3) adamantino;
4) graxo;	5) gorduroso;	6) resinoso;
7) translúcido;	8) barrento;	9) perolado;
10) sedoso;	11) acetinado;	12) brilhante.

Densidade:

A amostra é pesada no ar e dentro da água ou de um líquido específico. O valor encontrado não é sempre o mesmo para o mesmo mineral , havendo uma faixa de variação.

- Cor :

A cor é decorrente do elemento predominante no mineral, seja em forma de óxido ou de algum sal, dependendo ainda da quantidade de água de cristalização contida nos cristais:

A ametista tem a cor rosada ou roxa devido à presença do manganês. A cromita é marrom devido à presença de Fe (II) e óxido de cromo (III). A garnierita é esverdeada devido à presença do Níquel e assim por diante.

Com exceção da densidade, que é calculada, todas as outras características da amostra são determinadas graças à experiência prática do químico. Quando uma pessoa menos experiente realizar as mesmas observações, os resultados podem variar bastante devido à incerteza que cerca todos os ensaios.

Depois de todas as observações realizadas, determina-se qual o provável mineral. Porém, para comprovar a exatidão destas observações é necessária a análise química qualitativa.

3. Análise química qualitativa

A análise química qualitativa pode ser feita por via seca ou por via úmida. Os métodos por via úmida se dão pela manipulação das substâncias em solução, e se baseiam em transformações da substância problema em uma substância com características conhecidas que permitam sua identificação. A via úmida, por se tratar de reações entre soluções, requer que as substâncias ensaiadas sejam solubilizadas em água, quando solúveis nesta, e nos demais casos solubilizadas em ácidos diretamente ou após sofrerem fusão prévia. Na via úmida sabe-se que se produziu uma reação pela formação de um precipitado, por desprendimento de um gás ou pela mudança da cor.

Com a determinação do provável mineral, inicia-se a marcha analítica através da solubilização da amostra em um solvente específico que o químico estabelece graças a sua experiência. Em seguida, realiza-se uma série de reações químicas, de acordo com a formação ou não de precipitados e suas cores ou mudanças de cor das soluções. Ao final desta série de reações químicas, pode-se saber a composição química da amostra e conferir se os cátions encontrados realmente compõem a suposta amostra; ou seja, se o nome do mineral estabelecido inicialmente, realmente confere com o resultado final. O problema em questão utiliza este método.

ANEXO II

FuzzyCLIPS - Comandos Básicos

1. Introdução

A *shell* FuzzyCLIPS fornece os elementos básicos de um sistema especialista: (i) memória global de dados; (ii) base de conhecimento; (iii) máquina de inferência.

O texto a seguir serve como um mini manual para compreender o funcionamento desta *shell.* Neste texto algumas padronizações são feitas.

* comandos FuzzyCLIPS, são escritos em negrito: **assert**.
* construções FuzzyCLIPS são escritas na seguinte fonte: `construções.`
* as palavras em língua estrangeira são escritas da seguinte maneira: *case sensitive*.
* comandos da tela principal são escritos da seguinte maneira: ***tela principal***.

Todos os comandos do FuzzyCLIPS são escritos entre parênteses "( )" e os comentários são escritos sempre após um ponto e vírgula ";". O FuzzyCLIPS é *case sensitive*, isto é, distingue entre maiúsculo e minúsculo.

Cada construção do FuzzyCLIPS possue uma série de comandos próprios para sua manipulação. Sendo assim, tem-se comandos de ambiente, comandos para fatos, comandos para regras, comandos para classes, comandos para instâncias, comandos para variáveis, comandos para funções, etc. Tais comandos serão descritos a seguir.

2. Comandos Básicos

2.1 Fatos

Em FuzzyCLIPS, um fato corresponde a qualquer evento, situação ou objeto, que se quer armazenar na base de conhecimento.

2.1.1 Classificação

Os fatos podem ser classificados em simples e compostos. Esses por sua vez, são classificados em nomeado e não-nomeado.

Fatos simples possuem apenas um único campo, por exemplo, (cruzeiro).

Fatos compostos têm mais de um campo e podem ser nomeados ou não. Em um fato composto nomeado, a relação entre os campos é explicitada na representação do fato. Por exemplo, na declaração do fato (time-torcedor cruzeiro anita) fica explícito que o primeiro campo é um time e o segundo campo é um torcedor e dessa forma, deverá ser entendido pelas regas. Já em um fato composto não-nomeado, a relação não é explicitada, podendo ser interpretada pela regras de qualquer forma. Por exemplo, no fato (cruzeiro anita), a relação não é indicada e pode ser entendida de maneiras diferentes, tais como: "a empresa cruzeiro é presidida por anita" , "a cidade de cruzeiro tem uma moradora chamada anita" e até mesmo "o time cruzeiro tem uma torcedora chamada anita".

2.1.2 Comandos Básicos

a) Criação de fatos

Para inserir fatos em uma lista de fatos, usa-se o comando **assert**, cuja sintaxe é:

assert ((argumento))

Como exemplo para este comando, tem-se: **(assert (cruzeiro)),** onde cruzeiro é o argumento.

b) Verificação dos fatos que compõem a lista de fatos

Para verificar os componentes de uma lista de fatos, usa-se o comando **facts**, cuja sintaxe é:

(facts)

Quando este comando é acionado, obtem-se como resposta as seguintes linhas:

```
(facts)
f-0 (cruzeiro)
For a total of 1 fact.
```

Isto significa que o fato (cruzeiro) é o único da lista, e seu identificador é f-0. Se o FuzzyCLIPS for inicializado com **(clear)** ou **(reset),** a lista de fatos será zerada, e então, os fatos terão seus identificadores incrementados em um, à medida que forem sendo inseridos novamente na lista. Além disso, o comando **(reset)** incluirá um fato especial, **initial-fact**, que possuirá identificador igual a f-0. Este fato sempre é usado para iniciar as regras.

Se o usuário tentar inserir dois fatos com o mesmo nome na lista, isto não será permitido.

c) Extração de fatos da lista

Vários fatos podem ser extraídos da lista em uma única operação. A retirada dos fatos também remove todas as regras que dependem destes fatos para serem ativadas. Quando os fatos são removidos, os outros que permaneceram não têm seus identificadores alterados. O comando responsável pela extração de fatos é o **retract**, cuja sintaxe é :

(retract 1) - retira somente o fato f-1;
(retract 0 1) - retira os fatos f-0 e f-1;
(retract *) - retira todos os fatos que estão na lista.

d) Carregar fatos de um arquivo

A função **(loadfacts nome-do-arquivo)** carrega fatos que estão armazenados em um arquivo direto na lista de fatos. Ela pode ler arquivos criados pelo **(savefacts)** ou qualquer outro arquivo de texto ASCII.

e) Salvar fatos em arquivo

A função **(savefacts)** salva todos os fatos da lista em um arquivo.

2.2 Regras

Regra corresponde à forma básica de representação do conhecimento. As regras são compostas de antecedentes e consequentes. Os consequentes só são executados quando os antecedentes são satisfeitos. A forma básica de uma regra corresponde a:

SE
condição
ENTÃO
ação

As regras em FuzzyCLIPS são estruturadas da seguinte maneira:

```
(defrule time
     (time  cruzeiro)
⇒
     (assert (cor  azul))
)
```

A primeira linha corresponde ao nome da regra "time". Na segunda linha tem-se o antecedente da regra, que neste caso é constituído de um único fato (time cruzeiro). O símbolo "⇒ " corresponde ao início do *THEN*. A quarta linha representa o consequente da regra. Finalmente a quinta linha, contem o parêntese que fecha a construção (tal parêntese foi aberto no início da primeira linha.

Não pode haver regras com mesmo nome. Uma regra pode ter múltiplos antecedentes e/ou consequentes. Se todos os fatos do antecedente forem satisfeitos, a regra será ativada e colocada na agenda. Um programa encerra a execução quando não há mais nenhuma regra na agenda.

Quando várias regras estão na agenda, determina-se qual regra deve disparar. As regras são ordenadas de acordo com a prioridade ou saliência. Se a saliência da regra não é declarada, assume-se saliência igual a zero (saliência pode variar de -10.000 a +10.000).

2.2.1 Comandos Básicos

a) Visualização de uma regra

Para visualizar a estrutura de uma regra, usa-se o comando **ppderule**, cuja sintaxe é:

(ppderule nome-da-regra)

Caso seja necessário saber o nome da regra, usa-se o comando simples **(listdefrule),** que exibe o nome de todas as regras do programa em execução.

Para visualizar todas as regras usa-se o comando **(rules).**

b) Deleção de regras

Para deletar uma regra, usa-se o comando **(undefrule nome-regra).**

Se a regra a ser deletada estiver na agenda, este comando não será executado. Se "nome-regra" for igual a *, todas as regras serão deletadas.

c) *Breakpoint*

Se um *breakpoint* é colocado em uma dada regra, a execução encerrará antes de executar esta regra. Para inserir um *breakpoint*, usa-se o comando **(setbreak nome-regra).** Para removê-lo, usa-se o comando **(removebreak nome-regra).** Se nenhum argumento é fornecido, todos os *breakpoints* são removidos.

d) Declaração da saliência de uma regra

Para declarar a saliência, usa-se o seguinte comando: **(declare (salience valor))**. O valor pode variar de - 10.000 a + 10.000. A regra que tiver maior saliência dispara primeiro.

2.3 *Templates*

No FuzzyCLIPS, *templates* são construções usadas para definir os conjuntos difusos de uma determinada variável lingüística. Isto é feito através da construção **(deftemplate).** Como exemplo tem-se:

```
(deftemplate altura  ; variável linguística altura
     0  2 metros     ; universo de discurso desta variável
     (baixo (0 0)  (1.5 1) (2.0 0))
     (alto   (1.5 0) (1.8 1) (2.0 1))
)
```

No caso deste exemplo, os conjuntos difusos estão na forma de *singletons*. O usuário pode escolher entre as funções triangular, S, Z e Π, que já são pré-definidas pelo FuzzyCLIPS, ou pode criar suas próprias funções.

2.3.1 Comandos Básicos

a) Exibição do conteúdo de uma *template*

Para visualizar o conteúdo de uma *template*, usa-se o comando **(ppdeftemplate nome-da-template).**

b) Listagem do nome de todas as *templates* de um dado programa

Para esta tarefa, usa-se o comando **(listdeftemplates módulo-do-programa).** Se o módulo-do-programa não for determinado, será listado somente o nome das *templates* do módulo corrente. Se o módulo-do-programa for igual a *, o nome de todas as *templates* em todos os módulos será exibido.

c) Deleção de *templates*

O comando para deleção de *templates* é **(undeftemplate nome-template).** Se o *template* está em uso, este comando não será executado. Se nome-template for igual a *, todos os templates serão deletados.

2.4 Variáveis

O nome de uma variável local em FuzzyCLIPS é sempre escrito com um sinal de ? antes do nome, por exemplo, ?nome. Já as variáveis globais têm a seguinte sintaxe: **?*nome***, por exemplo, ?*idade*.

O FuzzyCLIPS envia uma mensagem de erro quando uma variável que não foi previamente declarada é usada. Para evitar problemas deste tipo, as variáveis globais devem ser definidas no início do programa através da construção **defglobal.** Por exemplo:

```
(defglobal
      ?*nome*   = " "
      ?*idade*   = 0
)
```

2.4.1 Comandos Básicos

a) Assinalando valores para uma variável

Para assinalar o valor para uma dada variável, usa-se o comando **bind**, cuja sintaxe é:

(bind nome-variável valor)

O nome da variável pode corresponder a uma variável global ou local, já o valor pode ser também uma expressão numérica, por exemplo:

(bind ?tamanho 9)

(bind ?tamanho (+ ?x ?y))

Este exemplo ilustra também como os operadores matemáticos são usados no FuzzyCLIPS.

3. O ambiente FuzzyCLIPS

O menu principal do FuzzyCLIPS possue uma série de opções que serão descritas à seguir.

3.1 *File*

Esta opção controla as operações com arquivos (carregar e salvar), através das subopções descritas a seguir.

a) ***Load Constructs*** *(Ctrl + L):* Carrega para a memória de trabalho todas construções contidas no programa a ser executado, ou seja, regras, fatos, *templates*, variáveis globais ou não, funções, etc.

b) ***Load Batch:*** Um arquivo *batch* permite o processamento interativo dos comandos do FuzzyCLIPS pela substituição das entradas padrão por conteúdo de um determinado arquivo. Qualquer comando ou função pode ser usado em um arquivo *batch*.

c) ***Load Binary:*** Carrega as construções armazenadas em um arquivo binário. O arquivo a ser carregado por este comando tem que ter sido criado pelo *Save Binary*. Carregar este tipo de arquivo é bem mais rápido do que carregar arquivos ASCII.

d) ***Turn Dribble on:*** Quando esta opção é selecionada, todas as informações correspondentes à execução de um programa são direcionadas para um arquivo, ou seja, todos os passos do processamento terão as suas entradas, saídas, tempo de execução e possíveis falhas, registradas neste arquivo. Caso esta opção não seja selecionada, as informações serão enviadas para a tela.

e) ***Save Binary:*** Salva todas as construções e todas as estruturas de dados a elas associadas (tais como fatos e instâncias). A ação de salvar também é bastante rápida para este tipo de arquivo, uma vez que, a representação *pretty print* de uma construção não é salva na imagem binária. As informações associadas às construções também não são salvas.

f) ***Editor (Ctrl + E):*** Carrega o arquivo referente ao editor de texto do FuzzyCLIPS.

g) ***Quit CLIPS:*** Esta opção sai do ambiente FuzzyCLIPS e retorna ao sistema operacional.

3.2 Edit

a) *Paste (Ctrl + U):* Insere o conteúdo que está na área de transferência.

b) *Complete (Ctrl + J):* Após digitar o início de um comando qualquer do FuzzyCLIPS, *Complete* fornece o restante da sintaxe correspondente para completar tal comando.

3.3 Execution

Corresponde aos comandos que controlam a execução de qualquer programa no ambiente FuzzyCLIPS.

a) ***Reset (Ctrl + U):*** Reseta o FuzzyCLIPS. Remove todas as regras que estão na agenda, todos os fatos das listas e todas as instâncias das classes. A seguir, reinicializa todos os valores das variáveis globais, armazena na memória os fatos estabelecidos pelo deffacts, cria todas as instâncias, estabelece que o módulo MAIN é o módulo atual.

b) ***Run (Ctrl + R):*** Inicia a execução de um determinado programa.

c) ***Step (Ctrl + T):*** Com esta opção, o usuário pode optar por acompanhar a execução de um programa passo a passo.

d) ***Watch (Ctrl + W):*** Esta opção permite que mensagens sejam exibidas quando certas operações são realizadas.

e) ***Options:*** opções para avaliação da saliência das regras e escolha da estratégia de resolução de conflitos.

f) ***Preferences*:** opções para ativar a exibição de *warnings* e escolha do incremento do disparo de regras.

g) ***Clear CLIPS:*** Remove todas as construções e todas as estruturas de dados associados (tais como **facts e instances**) do ambiente FuzzyCLIPS. Ele pode ser executado a qualquer hora, entretanto, certas construções não poderão ser deletadas enquanto estiverem em uso. Por exemplo: enquanto **deffacts** estão sendo resetados pelo comando **(reset),** não é possível removê-los usando o comando **(clear).**

3.4 Browse

a) ***Module Manager:*** exibe qual o módulo corrente.

b) ***Defrule Manager:*** exibe as regras que estão na agenda, oferecendo a opção para removê-las.

c) ***Deffacts Manager:*** exibe os fatos que estão na lista, oferecendo a opção para removê-los.

d) ***Deftemplate Manager:*** exibe os *templates* em uso, oferecendo a opção para removê-los.

e) ***Deffunction Manager:*** exibe as funções em uso, oferecendo a opção para removê-las.

f) ***Defglobal Manager***: exibe as variáveis globais do programa, oferecendo a opção para removê-las.

g) ***Defgeneric Manager:*** exibe as funções genéricas em uso, oferecendo a opção para removê-las.

h) ***Defclass Manager:*** exibe as classes em uso, oferecendo a opção para removê-las.

i) ***Definstances Manager:*** exibe as instâncias em uso, oferecendo a opção para removê-las.

j) ***Agenda Manager:*** exibe todo o conteúdo da agenda.

3.5 Window

Esta opção permite que algumas janelas sejam exibidas ou não durante a execução do programa.

a) ***facts window***: mostra os fatos que estão entrando ou saindo da lista de fatos.

b) ***agenda window:*** mostra o conteúdo corrente da agenda.

c) ***instances window:*** mostra as instâncias que estão sendo criadas nas classes.

d) ***globals window:*** exibe todas as variáveis globais.

e) ***focus window:*** exibe qual o módulo que está sendo usado.

f) ***all above:*** exibe todas as janelas acima citadas, em uma única tela.

g) ***none:*** não exibe as janelas citadas anteriormente.

h) ***clear dialog box:*** fecha todas as janelas que estão abertas, com excessão da janela principal.

3.6 Help

Fornece as informações quanto a sintaxe dos comandos.

a) ***About FuzzyCLIPS 6.02:*** apresenta algumas informações técnicas sobre o FuzzyCLIPS 6.02.

b) ***CLIPS Help:*** fornece a sintaxe e algumas informações adicionais sobre os comandos utilizados no FuzzyCLIPS.

c) ***Command Complete:*** fornece a sintaxe de um dado comando.

4. Operadores Lógicos , Aritméticos e de Comparação

4.1 Operadores Lógicos

Os operadores lógicos usados no FuzzyCLIPS são: **not, and, or**. A sintaxe para estes operadores é a seguinte:

(operador (fato 1) (fato 2)..... (fato n))

Exemplos: (or (cor azul (cor branca) (cor verde))

(not (cor azul) (cor branca) (cor verde))

(and (cor azul) (cor branca) (cor verde))

4.2 Operadores Aritméticos

Os operadores aritméticos usados em FuzzyCLIPS são: /, *, +, -. A sintaxe para estes operadores é a seguinte:

(operador variável 1 variável 2)

Exemplos: (+ ?var1 ?var2)

(/ (- ?var1 ?var2) ?var3)

4.3 Operadores de Comparação

Os operadores de comparação usados no FuzzyCLIPS são:

eq igual para variáveis ou valores de qualquer tipo (compara tipo e magnitude).
neq diferente para variáveis ou valores de qualquer tipo (compara tipo e magnitude).
= igual para tipos numéricos. Compara a magnitude.
<> diferente para tipos numéricos. Compara a magnitude.
>= maior ou igual a .
> maior que.
<= menor ou igual a.
< menor que.

Todas as funções de comparação, exceto **"eq" e "neq"** darão uma mensagem de erro se forem usadas para comparar números com não-números.

5. Representação dos conjuntos Difusos

5.1 Representação *Singleton*

Um *singleton* é representado no FuzzyCLIPS como um par $(x_i \ \mu_i)$. Um conjunto difuso A será representado como uma lista de *singletons*:

$$A = ((x_1 \ \mu_1), (x_2 \ \mu_2), (x_3 \ \mu_3)....)$$

Considerando a variável linguística **quantidade**, sua definição seria:

```
(deftemplate quantidade          ; declaração da variável linguística
     0 9                         ; universo de discurso
     (                           ; inicia as declarações
        (pouco (1 0) (2 0.3) (3 0.9) (4 1) (5 0.8) (6 0.5) (7 0))
                    ; conjunto difuso "pouco"
     )                           ; fim das declarações
)                                ; fim da estrutura
```

5.2 Representação das funções padrão

As funções padrão fornecidas pelo FuzzyCLIPS são: **S, Z, Π**. Os parâmetros destas funções devem ser escolhidos dependendo da aplicação. Como exemplo, tem-se:

```
(deftemplate Temperatura
     5 65 Celsius
     (
          (frio       (z 10 26))
          (ok         (PI 2 36))
          (quente     (s 37 60))
     )
)
```

5.3 Modificadores

O modificador deve ser declarado dentro da construção **deftemplate**, uma vez que sua função é modificar os graus de pertinência nos conjuntos difusos. A declaração de um modificador é feita da seguinte forma:

(nome do modificador nome da função)

onde "nome do identificador" representa o nome do modificador e "nome da função" representa o nome de uma função pré-definida do FuzzyCLIPS, ou uma **deffunction** definida pelo usuário. Como exemplo, tem-se:

```
(deftemplate velocidade  ; declaração da variável linguística
       0   160  km/h      ; universo de discurso
       (                ; início da declaração dos conjuntos difusos
     (devagar (30 1) (60 0))  ; este conjunto usa a notação
                              ;singleton
     (rápido   (50 0) (80 1)) ; este conjunto usa a notação
                              ;singleton
        )                     ; fim da declaração dos conjuntos
                              ;difusos
        (                     ; início da declaração dos
                              ;modificadores
     (muito    sqr)           ; modificador muito usando a função
                              ;sqr, que é pré-definida
     (pouco   minha-função)   ; modificador pouco usando uma
                              ;função criada pelo usuário através
                              ;de deffunction
         )                    ; fim da declaração de modificadores
)                             ; fim da construção
```

6. Defuzificação

O FuzzyCLIPS fornece dois métodos para defuzificação: Centro de Gravidade e Média dos Máximos. A sintaxe destes métodos é a seguinte:

(moment-defuzzify ?variável) **; centro de gravidade**
(maximum-defuzzify ?variável) **; média dos máximos**

Para ilustrar o processo de defuzificação, tem-se:

```
(defrule defuzificação
     ?f <-  (temperatura ?)    ; ? é usado para assegurar a
                               ;unificação do fato temperatura
⇒
     (bind ?t  (maximum-defuzzify ?f)    ;transfere o resultado da
                                         ;defuzificação da variável
                                         ;?f para a variável ?t
     (printout t "Temperatura é : " ?t crlf) ; imprime o resultado
)
```

7. Mensagens de Erro

Para auxiliar na depuração de um programa em FuzzyCLIPS, algumas mensagens de erro são enviadas ao usuário.

[AGENDA1] O valor da saliência não corresponde a um valor inteiro.

[AGENDA2] O valor da saliência está fora da faixa permitida.

[AGENDA3] Este erro ocorre enquanto a saliência de uma regra está sendo avaliada.

[ANALYSIS1] Erro que ocorre quando dois fatos ou instâncias são direcionados para a mesma variável de endereço.

[ANALYSIS2] Este erro ocorre quando uma variável destinada a armazenar um padrão, recebe um endereço de um outro padrão.

[ANALYSIS3] Este erro ocorre quando uma variável destinada a um campo simples recebe um campo composto.

[ANALYSIS4] Alguma variável foi referenciada sem ter sido definida anteriormente.

[ARGACCES1] Este erro ocorre quando uma função recebe menos argumentos ou mais argumentos do que ela realmente suporta.

[ARGACCES2] Este erro ocorre quando uma dada função não é capaz de abrir um arquivo.

[ARGACCES3] Este erro ocorre quando uma função recebe um número de argumentos inferior ao esperado.

[ARGACCES4] Este erro ocorre quando (i) uma função que espera por um número exato de argumentos, recebe um número diferente; (ii) uma função não recebe o mínimo de argumentos esperados; (iii) uma função recebe um número de argumentos maior que o esperado.

[ARGACCES5] Ocorre quando a função recebe um tipo de argumento diferente do esperado.

[ARGACCES6] Ocorre quando uma função solicita de uma outra função um tipo errado de argumento, tipicamente uma *string* ou símbolo, ao invés de número.

[BLOAD1] Ocorre quando não se consegue carregar uma construção de um arquivo binário.

[BLOAD2] Ocorre quando o arquivo a ser carregado não é um arquivo binário.

[BLOAD3] Ocorre quando o arquivo a ser carregado não é compatível com o padrão binário.

[BLOAD4] Ocorre quando o ambiente FuzzyCLIPS não pode executar o comando **clear** para carregar o arquivo.

[BLOAD5] Ocorre quando a imagem binária atual não pode sofrer a ação do comando **clear**, pois ainda estão em uso.

[BSAVE1] Ocorre quando há algum problema impedindo que um arquivo binário seja carregado.

[CLASSEXM1] Ocorre quando a função **slot-publicp** é aplicada a um *slot* que foi herdado. Esta função só pode ser aplicada aos *slots* definidos diretamente.

[CLASSFUN1] Ocorre quando uma função é atribuída a uma classe inexistente.

[CLASSFUN2] Ocorre quando o número de ativações simultâneas de diversas classes excede um limiar.

[CLASSPSR1] Ocorre devido ao fato de que apenas as classes concretas são reativas.

[CLASSPSR2] Ocorre quando tenta-se redefinir uma classe pré-definida pelo sistema.

[CLASSPSR3] Ocorre quando tenta-se redefinir uma classe, mas está ainda está sendo referenciada por outras classes.

[CLASSPSR4] Ocorre quando tenta-se redeclarar um atributo de uma classe.

[CLSLTPSR1] Ocorre quando tenta-se duplicar um *slot* em uma mesma classe.

[CLSLTPSR2] Ocorre quando tenta-se colocar mais de um tipo de acesso para o mesmo *slot*.

[CLSLTPSR3] Ocorre quando um campo simples recebe uma cardinalidade diferente de um.

[CLSLTPSR4] Ocorre quando um *slot* "*ready-only*" não apresenta um valor *default*.

[CLSLTPSR5] Ocorre quando um *slot* "*ready-only*" recebe acesso de escrita.

[CLSLTPSR6] Ocorre porque *slots* herdados não podem ser públicos.

[COMMLINE1] Ocorre quando está faltando um parêntese, uma constante ou uma variável global em alguma construção ou comando.

[COMMLINE2] Ocorre quando um comando está escrito faltando abrir ou fechar parênteses.

[CONSCOMP1] Ocorre quando um ponto (.) é usado no nome do arquivo, além do ponto da extensão.

[CONSTRUCT1] Ocorre quando o comando **clear** é chamado, mas algumas construções ainda estão em uso.

[CSTRCPSR1] Ocorre quando o comando **load** espera um parêntese à esquerda seguido

de um tipo de construção e estes tipos não são encontrados.

[CSTRCPSR2] Ocorre quando está faltando o nome de uma construção, ou o nome está escrito errado.

[CSTRCPSR3] Ocorre quando há um conflito entre o nome da construção e o tipo por ela definido.

[CSTRCPSR4] Ocorre quando tenta-se redefinir uma construção que está sendo usada por outra construção ou outro tipo de estrutura de dados.

[CSTRNCHK1] Erro de identificação, refere-se a uma faixa de mensagens indicando a violação do tipo, valor, faixa ou cardinalidade.

[CSTRNPSR1] Ocorre quando dois s*lots* apresentam conflito de atributos.

[CSTRNPSR2] Ocorre quando o valor mínimo de um atributo para uma faixa e os atributos de cardinalidade têm que ser menor ou iguais ao valor máximo do atributo para o atributo.

[CSTRNPSR3] Ocorre quando alguns atributos de *slots* excluem o uso de outros atributos do *slot*

[CSTRNPSR4] Ocorre quando os argumentos para um atributo não correspondem ao tipo esperado para aquele atributo.

[CSTRNPSR5] Ocorre quando atributos de cardinalidade são usados por *slots* que não estão com a palavra chave *multislot*

[DEFAULT1] Ocorre quando o valor para um *slot* de campo simples não é um valor de campo simples.

[DFFNXPSR1] Ocorre quando o nome de uma **deffunction** é igual ao de qualquer outra construção.

[DFFNXPSR2] Ocorre quando uma **deffunction** tem o mesmo nome de uma função do sistema, ou de uma função externa.

[DFFNXPSR3] Ocorre quando uma **deffunction** tem o mesmo nome de uma função genérica.

[DFFNXPSR4] Ocorre quando tenta-se redefinir uma **deffunction**, mas esta ainda está sendo executada.

[DFFNXPSR5] Ocorre porque uma **deffunction** não pode ter o mesmo nome que qualquer função genérica importada de outro módulo.

[EMATHFUN1] Ocorre quando um argumento passado para unificar uma função não pertence ao domínio.

[EMATHFUN2] Ocorre quando um argumento para uma função matemática extendida causa um *overflow*.

[EMATHFUN3] Ocorre quando um argumento de uma função trigonométrica causa singularidade.

[EVALUATN1] Ocorre quando uma variável local que não recebeu um valor através do comando **bind**, é acessada.

[EVALUATN2] Ocorre apenas quando um nome de função inválido é passado para uma rotina externa em C pelo **CLIPSFunctionCall**.

[EXPRNPSR1] Ocorre quando o nome da função é interpretado como símbolo pelo sistema.

[EXPRNPSR2] Ocorre quando está faltando uma constante, variável ou expressão em um comando.

[EXPRNPSR3] Ocorre quando o FuzzyCLIPS não reconhece o nome declarado para uma função.

[EXPRNPSR4] Ocorre quando o operador $ é usado no lugar indevido.

[FACTMNGR1] Ocorre quando tenta-se extrair fatos durante o processo de unificação.

[FACTRHS1] Ocorre quando um template que não existe é chamado.

[GLOBLDEF1] Ocorre quando uma variável que não foi definida é acessada.

[GLOBLPSR1] Ocorre quando uma variável global é referenciada sem estar definida.

[INSCOM1] Ocorre quando há um tipo indefinido em uma função.

[MEMORY1] Indica que não há memória suficiente para expandir estruturas externas.

[MEMORY2] Indica que há problema nas rotinas de gerenciamento de memória.

[MEMORY3] Indica que não há condições de alocar um bloco de memória maior que 32K.

[PRCCODE1] Indica que uma construção que não existe está sendo chamada.

[PRCCODE2] Indica que ocorreu um erro de avaliação durante o exame dos argumentos de uma **deffunction**.

[PRCCODE3] Indica que uma variável indefinida está sendo referenciada no antecedente de uma regra.

[PRCCODE4] Indica que a execução foi interrompida durante as ações de uma determinada construção.

[PRNTUTIL1] Ocorre quando FuzzyCLIPS não encontra o nome de um determinado item.

[PRNTUTIL2] Manda checar à sintaxe apropriada para uma determinada construção.

[PRNTUTIL3] Indica que aconteceu algum problema na estrutura interna do FuzzyCLIPS.

[PRNTUTIL4] Indica que uma determinada construção não podem ser deletada naquele instante.

[PRNTUTIL5] Indica que um determinado item já sofreu *parsing*.

[PRNTUTIL6] Indica que variáveis locais não podem ser acessadas por uma determinada função ou construção.

[PRNTUTIL7] Indica que ocorreu um erro relacionado a uma divisão por zero.

ANEXO III

FuzzyCLIPS - Exemplos de Programas

Para facilitar o aprendizado de quem quer trabalhar com FuzzyCLIPS, algums exemplos de programas serão descritos à seguir.

Exemplo 1: *fzCmplr.clp*

Este programa baseia-se na implementação de um exemplo fornecido pelo artigo "***A Compiler for Fuzzy Logic Controllers***' ", escrito por **P. Bonissone**, e publicado em *Fuzzy Eng. Human Friendly Systems*, volume 2, pp. 706 - 717.

Este programa não trata do problema dos compiladores como no artigo original, apenas calcula a saída dos resultados do controlador sobre uma faixa de valores de entrada para comparação. Após a comparação, escolhe-se a melhor entrada para o controlador. As saídas geradas por este programa são armazenadas no arquivo fzCmplr.dat.

O programa trabalha com duas variáveis linguísticas de entrada: temperatura e pressão; uma variável de saída: variação da válvula de controle; e utiliza o método do centro de gravidade para a defuzificação.

O programa está dividido em três partes: (i) definição dos conjuntos difusos associados às variáveis linguísticas de entrada; (ii) as regras para controlar a variação da válvula; (iii) regras para definir os valores de teste e produzir às saídas.

1. Conjuntos Difusos Associados às Variáveis Linguísticas

a) Variável temperatura:

O universo de discurso desta variável corresponde ao intervalo de [0 100] graus Celsius. Os conjuntos difusos são: baixa, média e alta. Não ocorre o uso de modificadores. Utiliza-se a representação Π.

```
(deftemplate temperatura
        0 100 Celsius
        ((baixa (PI 20 20))
        (media (PI 20 50))
        (alta   (PI  20 80))
        )
)
```

b) Variável pressão

O universo de discurso desta variável corresponde ao intervalo de [0 500] kPa. Os conjuntos difusos são: baixa, média, alta. Não ocorre o uso de modificadores. Utiliza-se a representação Π.

```
(deftemplate pressao
        0 500 kPa
        ((baixa (PI 100 100))
        (media (PI 100 250))
        (alta   (PI  100  400))
        ) )
```

c) Variável válvula

O universo de discurso desta variável corresponde ao intervalo de [0 1] unidades. Os conjuntos difusos são: muito-baixo, baixo, médio-baixo, médio, médio-alto, alto. Não ocorre o uso de modificadores. Utiliza-se a representação Π.

```
(deftemplate valvula
        0 1  unidades
        ((muito-baixa (PI 0.1 0.1))
         (baixa (PI 0.15  0.25))
        (medio-baixa (PI 0.1 0.4))
        (media   (PI  0.15  0.55))
        (media-alta (PI 0.1 0.75))
        (alta (PI 0.1 0.9))
        )
)
```

2.Regras que Trabalham com o Controle da Válvula

a) Regra temperatura baixa - pressão baixa.

SE
temperatura baixa AND pressão baixa
ENTÃO
coloque na lista de fatos: "Valvula alta".

Em FuzzyCLIPS, esta regra é representada por:

```
(defrule baixa-baixa
        (temperatura baixa)
```

(pressao baixa)

⇒

(assert (valvula alta))

As demais regras seguem a mesma estrutura.

3.Regras para Definir os Valores de Teste e Determinar a Saída

a) Regra **init**

Esta regra difere das demais pelo fato de não possuir antecedente. Este fato indica ao sistema que esta deve ser a primeira regra a disparar, dando início à execução do programa.

Estudando a regra init linha a linha, tem-se:

* **open** : é usado para abrir um arquivo. Este comando exige que arquivos que não tenham a extensão .clp sejam renomeados, a fim de serem tratados pelo FuzzyCLIPS. Sendo assim, na linha de comando: **(open "fzCmplr.dat" fzctl "w")**, tem-se que o arquivo "fzCmplr.dat" é renomeado para fzctl. Além disto, este arquivo recebe o atributo de escrita ("w"), fazendo com que dados possam ser gravados nele.

* **format**: formata a saída dos dados que estão armazenados no arquivo, de maneira que se tenha a seguinte forma:

Temperatura	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95
Pressao																			
10	x																		

sendo que no local onde está o x, fica o valor correspondente à variação da válvula, quando a temperatura é 5 e a pressão é 10, e assim por diante.

* **assert**: irá acrescentar à lista de fatos, os seguintes fatos:

Temperatura igual a (PI 0.5 5)

Pressão igual a (PI 0.5 10)

b) Regra **unifica-temperatura-valvula**

* **declare** : declara a saliência da regra, ou seja, a prioridade de disparo da regra.

* **?t <-**: transfere para a variável ?t , o conteúdo correspondente ao conjunto difuso formado para suportar a temperatura fornecida pelo usuário.

?th <-: transfere para a variável ?th, conteúdo correspondente ao conjunto difuso formado para suportar a variação da válvula.

* **(crisp temp (moment-defuzzify ?t))**: corresponde à composição do seguinte fato:

(valor crisp de temperatura é igual ao valor da defuzificação de ?t pelo método do centro de gravidade)

* **retract**: retira da lista de variáveis, as variáveis ?t e ?th.

c) Regra **proxima-temperatura**

* **?p**: transfere para a variável ?p o conjunto difuso formado para suportar a variação da pressão.

* **format**: formata os dados do arquivo para uma precisão de duas casas decimais para a válvula.

* **if**: avalia a seguinte condição: se ?t > t, então...

* **bind**: assinala o valor 5 para a variável ?t.

* **integer**: considera somente a parte inteira de um número.

d) Regra **proxima-pressao**

* **close**: fecha o arquivo fzctl.

* **halt**: encerra a execução do programa, similar ao ***exit*** .

A listagem completa do programa é exibida à seguir.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;fzCmplr.clp;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; definicao dos conjuntos difusos

(deftemplate temperatura
    0 100 C
    ((baixa (PI 20 20))
     (media (PI 20 50))
     (alta  (PI 20 80))
    ) )

(deftemplate pressao
    0 500 kPa
    ((baixa (PI 100 100))
     (media (PI 100 250))
     (alta  (PI 100 400))
    ))

(deftemplate valvula
    0 1 unidades
    ((muito-baixa (PI 0.1  0.1))
     (baixa      (PI 0.15  0.25))
     (media-baixa (PI 0.1  0.4))
     (media      (PI 0.15  0.55))
     (media-alta  (PI 0.1  0.75))
     (alta       (PI 0.1  0.9))
    ))

;;; definicao das regras que trabalham com a variacao da valvula (9 regras)

(defrule baixa-baixa
    (temperatura baixa)
    (pressao baixa)
=>
    (assert (valvula alta))
)

(defrule baixa-media
    (temperatura baixa)
    (pressao media)
=>
    (assert (valvula media))
)

(defrule baixa-alta
    (temperatura baixa)
    (pressao alta)
=>
    (assert (valvula media-baixa))
)

(defrule media-baixa
    (temperatura media)
    (pressao baixa)
=>
    (assert (valvula media-alta))
)

(defrule media-media
    (temperatura media)
    (pressao media)
=>
    (assert (valvula media-baixa))
)

(defrule media-alta
    (temperatura baixa)
    (pressao alta)
=>
    (assert (valvula baixa))
)

(defrule alta-baixa
    (temperatura alta)
    (pressao baixa)
```

```
=>
    (assert (valvula media-baixa))
)

(defrule alta-media
    (temperatura alta)
    (pressao media)
=>
    (assert (valvula baixa)) )

(defrule alta-alta
    (temperatura alta)
    (pressao alta)
=>
    (assert (valvula muito-baixa))
)

;;; regras que controlam a determinacao dos valores de teste e produz as saidas

(defrule init
=>
    (open "fzCmplr.dat" fzctl "w")
    (format fzctl "Temperatura 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
                   80 85 90 95%nPressao%n 10")
    (assert (temperatura (PI 0.5 5))
        (pressao (PI 0.5  10))
    ) )

(defrule unifica-temperatura-valvula
    (declare
        (salience -100)
    )
    ?t <- (temperatura ?)
    ?th <- (valvula ?)
=>
    (assert (crisp temperatura (moment-defuzzify ?t)))
    (assert (crisp valvula (moment-defuzzify ?th)))
    (retract ?t ?th)
)

(defrule proxima-temperatura
    ?ct <- (crisp temperatura ?t)
    ?p  <- (pressao ?)
    ?cth <- (crisp valvula ?th)
=>
    (format fzctl "%5.2f" ?th)
    (if >= ?t 95)
      then
        (bind ?t 5)
        (assert (crisp pressure (moment-defuzzify ?p)))
        (retract ?p)
      else
        (bind ?t +5 (integer (+ 0.1 ?t))))
    )
        (assert (temperatura (PI 0.5 ?t)))
        (retract ?ct ?cth)
)

(defrule proxima-pressao
    ?cp <- (crisp pressao ?p)
=>
    (retract ?cp)
    (bind ?p (+10 (integer (+ 0.1 ?p))))
    (if (> ?p 490)
        then
          (format fzctl "%n%n")
          (close fzctl)
          (halt)
        else
          (format fzctl %n %3d " ?p)
          (assert (pressao (PI 0.5 ?p)))
    ) )
```