UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO
E AUTOMAÇÃO INDUSTRIAL

SISTEMA DE MANIPULAÇÃO DE CONJUNTOS E RELAÇÕES NEBULOSAS E PROGRAMAÇÃO EM LÓGICA NEBULOSA - PROLOG NEBULOSO.

Por : Ivan Rizzo Guilherme
Orientador : Prof. Dr. Márcio Luiz de Andrade Netto

Tese de Mestrado Apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas.

Agosto - 1990

Este Trabalho contou com o apoio financeiro do

Conselho Nacional de Desenvolvimento
Científico e Tecnológico - CNPQ

Dedico este trabalho a

a meus pais,

irmãs e

sobrinhos.

Agradecimentos

Ao Prof. Dr. Márcio Luiz de Andrade Neto, pela confiança, pelo incentivo, por dar-me liberdade na busca das soluções, pela segura orientação. Fatores fundamentais para a realização deste trabalho e para minha formação.

Ao Prof. Dr. Armando Freitas Rocha, professor adjunto do Departamento de Fisiologia e Biofisica, pelas oportunas sugestões e discussões, que influênciaram este trabalho e minha formação.

A Universidade de Campinas por dar-me a oportunidade de fazer este trabalho.

Ao Centro de Informática em Saúde, em especial ao seu coordenador Prof. Dr. Daniel Sigulem pelo imenso apoio dado para a realização deste trabalho.

Ao pesquisador Ricardo José Machado, do Centro Científico Rio - IBM Brasil.

Ao Agostinho, Ivani, Ivana e Monica pelo valiosissimo auxílio na correção do texto da monografia.

Aos amigos alunos do curso de pós graduação da Faculdade de Engenharia Elétrica.

Aos amigos do Centro de Informática em Saúde.

Ao meu primo Eduardo por acolher-me no "Mukifo" e por mantê-lo um ambiente agrádavel.

As pessoas que direta ou indiretamente auxiliaram no decorrer deste trabalho.

RESUMO

Este trabalho consiste da discussão e implementação de ferramentas de programação que operem Conjuntos Nebulosos e a Lógica Nebulosa.

É demonstrado e discutido um sistema composto de um conjunto de rotinas voltadas para a Manipulação de Conjuntos e Relações Nebulosas.

Alguns sistemas de programação baseados em lógica nebulosa, denominados de PROLOG-nebulosos, são analisados.

Finalmente descreve-se a implementação de um PROLOG-nebuloso que possui as mesmas características da linguagem PROLOG.

ABSTRACT

This work consists of a discussion and implementation of programming tools that operate the Fuzzy Sets Theory.

A system composed of a set of routines specialized on manipulating of Fuzzy Sets and Fuzzy Relations is shown and discussed.

Some system for Fuzzy Logic Programming, known as Fuzzy PROLOG, are analysed.

Finally, an implementation of a Fuzzy PROLOG is describe that has the same characteristic of the PROLOG language.

ÍNDICE

Simbologia Utilizada:

$\in$	Pertence.
$\notin$	Não Pertence.
$\subset$	Contido.
$\cap$	Interseção entre Conjuntos discretos.
$\cup$	União entre Conjuntos discretos.
.	Operador Produto.
$\mp$	Operador Soma Booleana.
$\tilde{A}$	Conjuntos nebulosos são representados por letras maiúsculas com ~.
μ	Grau de Pertinência.
$\wedge$	Interseção entre Conjuntos nebulosos.
$\vee$	União entre conjuntos nebulosos.
R	Todas Relações Nebulosa é Representa por Letras Itálicas Maiúscula.
$\circ$	Utilizado para representa a composição de relações.
$\neg$	Operador nebuloso "não".
&	Operador nebuloso "e".
$\vee$	Operador Nebuloso "ou".
$\rightarrow$	Consequência.

CAPÍTULO 1

INTRODUÇÃO

A ciência tem buscado meios que permitam ao homem vencer as dificuldades e limitações que lhe são impostas, e a compreender ele próprio e seu habitat.

Certamente uma de suas principais buscas tem sido o entendimento dos processos que envolvam o seu próprio funcionamento. Como consequência, tem crescido nos últimos anos o interesse da utilização do conhecimento do funcionamento humano na construção de máquinas que possam substituir o homem em atividades mecânicas e insalubres, e que possam liberá-lo para as atividades intelectuais. Estas pesquisas possibilitaram o desenvolvimento de muitas áreas como a Robótica, Visão, Reconhecimento de Padrões e Inteligência Artificial (em seu sentido amplo).

Nos últimos 60 anos houve um grande avanço nos estudos da inteligência humana (biologia, psicologia e medicina) que aliado ao aparecimento dos computadores geraram novas áreas de pesquisas. Dentre estas áreas, estão a Inteligência Artificial e Redes Neurais, que consistem em criar máquinas e programas que correspondam ao comportamento da inteligência humana.

Estas duas áreas, embora tenham o mesmo objetivo (modelar o comportamento da inteligência humana), ainda não possuem correspondência no plano teórico devido a conceituação inicial das suas pesquisas. Na Inteligência Artificial considera-se que a modelagem da inteligência humana deve ser feito atráves da manipulação simbólica das informações. Na área de Redes Neurais busca-se construir modelos distribuídos de máquinas e programas na qual a unidade básica de processamento é uma estrutura que possui as mesmas característas dos neurônios. Portanto a área está estruturada em torno de conceitos da fisiologia, matemática e da estatística, que são utilizados na modelagem do neurônio e dos outros mecanismos do cérebro.

O contexto deste trabalho é a Inteligência Artificial. Entretanto, pode-se notar no decorrer do mesmo, que os mecanismos disponíveis são bastante restritos e não correspondem ao processo de raciocínio humano, levando-nos para modelos de redes neurais que parece ser o destino atual das pesquisas nesta área.

Os modelos criados na I.A. são caracterizados por sistemas que possuem uma base de conhecimento (onde estão representados os símbolos) e um mecanismo de inferência para operar sobre os mesmos. Portanto, o primeiro passo consiste em

representar o conhecimento na base de dados. Como as informações em nossa volta são parciais, imprecisas, inconsistentes e incompletas, tornam-se necessário mecanismos de representação e de inferência com dados parciais e incompletos, tal como no raciocínio humano.

Há várias abordagens para a manipulação e inferência de dados incertos: os modelos baseados em probabilidade; os modelos baseados na teoria dos fatores de certeza (Dempster-Shafer); a teoria dos conjuntos nebulosos, etc..

Modelos baseados em probabilidade são largamente utilizados. Entretanto, estudos tem mostrado que estes conceitos probabilísticos não são processados naturalmente por especialístas ([STEF87 E SAGE87]) e os dados obtidos durante a aquisicão de conhecimento e processados durante o raciocínio geralmente não obedecem à complementaridade, que é uma das regras básicas da teoria das probabilidades.

Abordaremos neste trabalho a teoria dos conjuntos nebulosos como mecanismo para a representação e inferência sobre as informações.

Podemos verificar na literatura que a teoria dos conjuntos nebulosos tem se mostrado útil para a modelagem de sistemas inteligentes. Grande parte das implementações computacionais desses modelos são feitas através das ferramentas de programação tradicionais e consistem apenas de programas para a solução de um problema específico.

Abordaremos a construção de ferramentas de programação que facilitem o desenvolvimento de sistema inteligentes baseados na teoria dos conjuntos nebulosos.

O passo inicial consistiu em estudar a teoria dos conjuntos nebulosos. Baseado nesse estudo verificou-se ser interessante a construção de uma ferramenta computacional na qual pudéssemos representar conjuntos e relações nebulosas, e que contivesse implementado as operações mais importantes desta teoria.

Paralelamente verificou-se na literatura que as primeiras ferramentas de programação FDTS ([UMAN78]) e FRIL ([BALD84 e ZHOW84]) também possuem estas características.

A primeira característica deste sistema é possuir mecanismo para representação de conjuntos e relações nebulosas. Podemos construir conjuntos, relações, e utilizá-los para representar, por exemplo, variáveis linguísticas como "alto", "jovem", etc..Através dos conjuntos e relações podemos efetuar a representação de informações parciais e imprecisas.

As unidades básicas de manipulação são conjuntos e relações nebulosas. As operações de manipulação são funções em que os parâmetros são as unidades básicas (conjuntos e relações nebulosas).

Podemos utilizar esta ferramenta em problemas que operem ou utilizem

conjuntos e relações nebulosas discretas. Podemos implementar, utilizando as operações definidas, mecanismos de inferência, como, por exemplo, a teoria da possibilidade ([ZADEH88])

Este sistema, por operar somente sobre as unidades básicas, causa alguns inconvenientes, pois quando necessitamos obter a informação de apenas um elemento de um conjunto (ou relação), temos que efetuar a operação sobre todos os elementos do conjunto (ou da relação), gastando tempo desnecessário de processamento. Outra necessidade é a criação de uma interface amigável, que permita ao usuário maior facilidade de comunicação e de programação. Uma das derivações deste tipo de ferramenta são sistemas conhecidos como bases de dados nebulosas, que podem ser enquadradas como bases de dados inteligentes ([BALD84 e ZHOW84] e [ZEMA85 e KAND85]).

As limitações das primeiras ferramentas e o fato das ferramentas de programação da I. A. possuirem duas características: meios para representação simbólica e inferência sobre simbolos, fizeram com que as ferramentas acima fossem deixadas de lado.

Inicialmente tentou-se empregar a linguagem PROLOG, largamente utilizada em I.A, para construir sistemas que permitissem a representação das informações imprecisas (conjuntos e relações nebulosas) e meios para operar sobre as mesmas (lógica nebulosa).

Devido aos avanços teóricos na lógica nebulosa e com a definição do princípio de resolução nebuloso, foi possível definir um PROLOG-nebuloso.

Os PROLOG-nebulosos implementados (FUZZY PROLOG [MUKA87 et alli], FPROLOG [MART87 et alli]) e o implementado neste trabalho possuem as mesmas características sintáticas e semânticas do PROLOG convencional. Os PROLOG-nebulosos se diferenciam do PROLOG por possuírem um princípio de resolução nebuloso e mecanismos para representação de informações parciais e incompletas através de conjuntos e das relações nebulosas.

Os PROLOG-nebulosos são baseados nos operadores tradicionais da lógica nebulosa (máximo e mínimo) e nos predicados nebulosos de primeira ordem. Devido a lógica nebulosa suportar a lógica clássica, os PROLOG-nebulosos podem reutilizar os programas dos PROLOG convencionais, desde que escritos na sintaxe correspondente.

O aspecto declarativo facilita a construcão de programas e torna fácil a compreensão dos mesmos. Ao contrário das ferramentas descritas anteriormente, que manipulam somente conjuntos e relações nebulosas, a obtenção de informações contidas na base é feita sobre cada elemento, evitando processamento desnecessário.

O PROLOG-nebuloso não é uma ferramenta de programação capaz de operar em todas as aplicações da teoria dos conjuntos nebulosos em I. A. . Isto porque, com os

operadores máximo e mínimo e com os algoritmos do PROLOG, não podemos, por exemplo, construir mecanismos de raciocínio parcial e aproximado.

Com o aparecimento de novos operadores nebulosos, propostos para permitir uma melhor representação do comportamento do raciocínio humano (raciocínio aproximado e raciocínio parcial), tem-se verificado que os mecanismos do PROLOG são restritos. Como o problema se restringe ao modo de funcionamento do PROLOG, a solução é implementar um mecanismo mais complexo, onde tenhamos, por exemplo, um nível de decisão, ou utilizar um outro paradígma de programação, como, por exemplo, orientado por objetos.

A sequência em que este trabalho transcorreu é a mesma em que os trabalhos desta área foram desenvolvidos. Inicialmente construiram-se sistemas para manipulação de estruturas (conjuntos e relações) nebulosas, e em seguida estudou-se a implementação de ferramentas baseadas no PROLOG (PROLOG-nebuloso). Finalmente como consequência dos dois modelos, discute-se a nível teórico uma nova ferramenta de programação baseada no PROLOG, que opera um cálculo aditivo e que possui maior capacidade para a manipulação de informação parcial e imprecisa e formas de raciocínio parcial e aproximado.

No capítulo 2 efetuamos a descrição dos aspectos teóricos da teoria dos conjuntos e relações nebulosas. Discute-se a lógica nebulosa, baseada nos operadores mínimo e máximo, e descreve-se um operador nebuloso aditivo. Muito dos conceitos descritos serão implementados nas ferramentas descritas nos capítulos posteriores.

No capítulo 3 descreve-se a implementação da ferramenta de programação para manipulação dos conjuntos e das relações nebulosas. Descreve-se sucintamente as funções e algumas de suas aplicações.

No capítulo 4 discute-se como criar ferramentas de programação baseadas em lógica nebulosa utilizando o paradigma lógico. Inicialmente mostra-se como implementar no PROLOG convencional predicados que operem a lógica nebulosa e a representação do conhecimento impreciso. A seguir descreve-se modelos de PROLOG-nebulosos que se caracterizam por associar ao processo de resolução a lógica nebulosa baseada nos operadores max-min e possuir meios para representar os dados imprecisos. Descreve-se a proposta de uma ferramenta de programação baseada no PROLOG e que trabalha com um único operador aditivo.

E para finalizar no capítulo 5 descreve-se a implementação de um PROLOG-nebuloso, o qual é qualificado como convencional, por possuir grande parte das características do PROLOG convencional.

CAPÍTULO 2

ASPECTOS TEÓRICOS

Neste capítulo define-se a teoria dos conjuntos nebulosos. Inicialmente efetua-se a revisão da teoria dos conjuntos ordinários, a fim de permitir uma melhor compreensão da teoria dos conjuntos nebulosos. Serão definidos conjunto, grafo e relação nebulosa e as operações de manipulação. Parte das operações definidas estão implementadas no sistema de manipulação de conjuntos e relações nebulosas que é descrito no próximo capítulo. É descrito o conceito de variáveis linguísticas e formas de gerar conjuntos nebulosos e variáveis linguísticas. Serão descritas ampliações do conceito de conjunto e relação nebulosa. Define-se a lógica nebulosa conceituada em [ZADE65] e uma de suas mais importantes variações, a teoria da possilidade, e descreve-se um cálculo concebido para permitir a aditividade e raciocínio parcial e aproximado. Os sistemas que manipulam esses conceitos lógicos serão visto no capítulo 4.

As principais referências bibliograficas utilizadas neste capítulo são: [KAUF75] utilizado nas definições teoricas de conjuntos e relações nebulosas, e operações sobre conjuntos e relações nebulosas; [NEGO75 e RALE75], [ZADE88] e [ZADE65] utilizado nas definições de lógica nebulosa e teoria da possibilidade; [LEE72] utilizado na definição da consequência lógica e do princípio de resolução nebuloso; e [ROCH89 et alli] utilizado para descrever o modelo aditivo.

2.1 REVISÃO DA NOÇÃO DE PERTINÊNCIA E CONJUNTOS ORDINÁRIOS

Seja E um conjunto e A um subconjunto de E, denotado pela simbologia: $A \subset E$.

Para indicarmos que um elemento x de E é um membro do conjunto A utilizamos a simbologia

$$x \in A$$

Pode-se definir uma função de pertinência, a qual denotaremos por $\mu_A(x)$, cujo valor em {0,1}, indica se x é ou não membro de A.

$$\mu_A(x) = \begin{array}{ll} 1 & \text{se } x \in A \\ 0 & \text{se } x \notin A \end{array}$$

Como exemplo considere um conjunto finito

$$E=\{ x_1, x_2, x_3, x_4, x_5\} \qquad A=\{ x_2, x_3, x_5\}$$

Os valores da função de pertinência no conjunto A são

$$\mu(x_1)=0, \quad \mu(x_2)=1, \quad \mu(x_3)=1, \quad \mu(x_4)=0, \quad \mu(x_5)=1.$$

Abaixo temos a representação simplificada do conjunto A, onde o primeiro elemento pertence a E e o segundo é o valor do grau de pertinência.

$$A=\{(x_1,0), (x_2,1), (x_3,1), (x_4,0), (x_5,1)\}.$$

Sobre os conjuntos ordinários pode-se aplicar as propriedades da álgebra booleana.

Seja o conjunto A_c, o complemento de A em relação ao conjunto E, e os símbolos $\cap$ e $\cup$ denotando interseção e união entre conjuntos. São válidas as seguintes propriedades:

$$A \cap A_c = 0, \qquad A \cup A_c = E.$$

Se $x \in A$ então $x \notin A_c$, pode-se afirmar que se a função de pertinência é $\mu_A(x)=1$ então $\mu_{A_C}(x)=0$. Pode-se definir o complemento do conjunto A (denotado por A_C) como:

$$A_C=\{(x_1,1),(x_2,0),(x_3,0),(x_4,1),(x_5,0)\}.$$

Dado dois conjuntos A e B, pode-se definir a operação interseção $A \cap B$ e sua função de pertinência como:

$$\mu_{A\cap B}(x) = \begin{array}{ll} 1 \text{ se} & x \in A \text{ e } x \in B \\ 0 \text{ se} & x \notin A \text{ ou } x \notin B \end{array}$$

Podemos criar a operação produto (operador multiplicacação denotado por '.') e a sua função de pertinência é definida da seguinte forma

$$\mu_{A.B}(x) = \mu_A(x) \cdot \mu_B(x)$$

$$\mu_{A.B}(x) = \begin{array}{ll} 1 & \text{se } x \in A \text{ e } x \in B \\ 0 & \text{se } x \notin A \text{ ou } x \notin B \end{array}$$

Deve-se salientar que as operações "." e "$\cap$" descritas acima possuem função de pertinência igual.

A função de pertinência da operação união (denotada pelo operador $\cup$) é definida da sequinte forma.

$$\mu_{A\cup B}(x) = \begin{array}{ll} 1 & \text{se } x \in A \text{ ou } x \in B \\ 0 & \text{se } x \notin A \text{ e } x \notin B \end{array}$$

A operação soma booleana é denotada pelo operador '$\bar{+}$', e uma tabela com os valores de sua função de pertinência é mostrada abaixo. Pode-se verificar que os valores de pertinência das operações "$\cup$" e "$\bar{+}$" são iguais.

$\bar{+}$	0	1
0	0	1
1	1	1

As operações união e interseção correspondem aos conectivos "ou" e "e" da lógica booleana.

2.2 CONCEITO DE SUBCONJUNTO NEBULOSO

Em nossa vida diária subjetivamente manipulamos dados nebulosos. Quando utilizamos conceitos tais como "alto", "perto" "aproximadamente", estamos intuitivamente manipulando conjuntos nebulosos que são associados a estes conceitos.

Um conjunto nebuloso consiste de elementos, que estão contidos no conjunto do universo do discurso, e seu respectivo grau de pertinência. O grau de pertinência de um elemento num conjunto nebuloso é dado por uma função de pertinência.

Considere o subconjunto A, contido em E, definido anteriormente. Os elementos de E podem pertencer ou não a A. Consequentementea função de pertinência toma valores em {0,1}.

Imagine agora a função de pertinência tomando valores no intervalo [0,1]. Pode-se então afirmar que, um elemento x_i de E não é membro de A ($\mu_A(x_i) = 0$), é menos membro de A ($\mu_A(x_i) = 0.25$, próximo de zero), é mais ou menos membro de A ($\mu_A(x_i) = 0.5$, nem tão próximo de 0 e nem tão próximo a 1), pertencer bastante a A ($\mu_A(x_i) = 0.8$, próximo a 1), e finalmente ser membro de A ($\mu_A(x_i) = 1.0$).

A representação de um conjunto nebuloso é feita da seguinte forma:

$$\hat{A} = \{(x_1 | \ 0,2), (x_2 | \ 0.0), (x_3 | \ 0.3), (x_4 | \ 0.4), (x_5 | \ 0.5)\}$$

onde x_i é o elemento do conjunto referência E, e o número após a barra é o valor da função característica ou função de pertinência. A forma de representação descrita acima é a adotada na literatura, e será adotada neste texto.

A seguir definiremos um conjunto nebuloso formalmente.

Seja E o universo do discurso, enumerável ou não. Seja x um elemento genérico de E e M o conjunto de pertinência. Então um subconjunto nebuloso Ã de E é caracterizado por uma função de pertinência

$$\mu_{\tilde{A}} : E \rightarrow M,$$

que associa com cada elemento x de E um número $\mu_{\tilde{A}}(x)$, contido em M, que representa o grau de pertinência de x em Ã. O conjunto nebuloso é representado por:

$$\{(x \ / \ \mu_{\tilde{A}}(x)), \ | \ x \in E \};$$

A função de pertinência pode ser representada por:

$$x \xrightarrow[\mu_{\tilde{A}}]{} M.$$

Normalmente o conjunto M é dado pelo intervalo [0,1]. Entretanto podemos variar o conjunto M. Se tomarmos M = {0,1}, passamos a ter um subconjunto ordinário. Pode-se verificar facilmente que os conjuntos ordinários são subconjuntos dos conjuntos nebulosos. Como a noção de subconjunto nebuloso está ligada á noção de conjunto, consequentemente seu estudo pode ser feito usando-se estruturas matemáticas já definidas. Denotaremos conjuntos nebulosos por uma letra maiúscula com o acento til(por exemplo Ã).

2.2.1 OPERAÇÕES SOBRE CONJUNTOS NEBULOSOS

Seja E um conjunto e associado a ele um conjunto de pertinência M=[0,1], sejam Ã e Õ dois subconjuntos nebulosos de E. Utilizando esses conjuntos podemos definir as seguintes propriedades:

Inclusão: Dizemos que Ã esta incluso em Õ se

$$\forall \ x \in E : \mu_{\tilde{A}}(x) \leq \mu_{\tilde{O}}(x)$$

Esta operação será denotada por $\tilde{A} \subset \tilde{O}$. Chamaremos de inclusão estrita, quando pelo menos uma relação é estrita (<) e denotaremos por

$$\tilde{A} \subset\subset \tilde{O}.$$

Igualdade: Dizemos que Ã e Õ são iguais se e somente se

$$\forall \ x \in E : \mu_{\tilde{A}}(x) = \mu_{\tilde{O}}(x) \quad \text{e denotamos por } \tilde{A}=\tilde{O}.$$

Se pelo menos um elemento x de E não satisfaz a igualdade $\mu_{\tilde{A}}(x)=\mu_{\tilde{O}}(x)$, dizemos então que Ã e Õ são diferentes, e denotamos por:

$$\tilde{A} \neq \tilde{O}.$$

Complemento: Os subconjuntos Ã e Õ são complementares se

$$\forall \ x \in E : \mu_{\tilde{A}}(x) = 1 - \mu_{\tilde{O}}(x)$$

Denotados por $\tilde{O} = \tilde{A}_c$. Deve-se notar que o complemento é definido para M=[0,1]. Podemos estender para outros conjuntos de pertinência, desde que usando definições apropriadas.

Interseção: A interseção entre dois subconjuntos nebulosos Ã e Õ, é definida como o menor subconjunto que contém elementos de Ã e Õ, é dado por:

$$\forall x \in E : \mu_{\tilde{A} \wedge \tilde{O}}(x) = \text{MIN}(\mu_{\tilde{A}}(x), \mu_{\tilde{O}}(x)) \text{ e denotado por } \tilde{A} \wedge \tilde{O}.$$

Este conceito permite introduzir o operador "e" nebulosos simbolizado por "ë". Para exemplificar: dado o conjunto nebuloso Ã de números muito próximos a 5 e o subconjunto nebuloso de números muito próximos a 10, através da operação "ë" obtemos um conjunto nebuloso dos números muito próximos a 5 "ë" 10. A conjunção "ë", será denotada por "e"; quando necessário, colocaremos o til.

União : A união entre subconjuntos nebulosos Ã e Õ, é definida como o maior subconjunto nebuloso contendo elementos de Ã ou Õ, é dado por:

$$\forall x \in E : \mu_{\tilde{A} \vee \tilde{O}}(x) = \text{MAX}(\mu_{\tilde{A}}(x), \mu_{\tilde{O}}(x)) \text{ e denotado por } \tilde{A} \vee \tilde{O}.$$

Este conceito permite introduzir o operador "ou/e" nebulosos simbolizado por "õu/ë". Para exemplificar, dado o conjunto nebuloso Ã de números muito próximos a 5 e dado o subconjunto nebuloso de números muito próximos a 10, através da operação "õu/ë" obtemos um conjunto nebuloso dos números muito próximos a 5 "õu/ë" 10. A conjunção "õu/ë" será escrita como ou/e, quando não houver risco de interpretação.

Soma disjuntiva : A operação soma disjuntiva entre dois subconjuntos nebulosos Ã e Õ é definida utilizando as operações união e interseção descritas acima.

$$\tilde{A} \oplus \tilde{O} = (\tilde{A} \wedge \tilde{O}) \vee (\tilde{A}_c \wedge \tilde{O})$$

Esta operação corresponde ao "ou disjuntivo" nebuloso, representado por "õu". Em alguns casos em que não exista risco de interpretação denota-se por "ou".

Diferença : A diferença é definida pela seguinte relação.

$$\tilde{A} - \tilde{O} = \tilde{A} \wedge \tilde{O}_c$$

onde $\tilde{O}_c$ é o complemento do conjunto nebuloso Õ.

Distância de Hamming: Usa-se o cálculo da distância de Hamming como medida de semelhança entre conjuntos nebulosos. É definida por

$$d(\tilde{A},\tilde{O}) = \sum_{i=1}^{n} | \mu_{\tilde{A}}(x_i) - \mu_{\tilde{O}}(x_i) |$$

quando $\mu_{\tilde{A}}(x_i)$, $\mu_{\tilde{O}}(x_i) \in [0,1]$, i=1..n, onde n é o número de elementos do conjunto. E o domínio é dado por:

$$0 \leq d(\tilde{A},\tilde{O}) \leq n.$$

Distância Euclidiana ou quadrática : É definida pela seguinte fórmula:

$$e(\tilde{A},\tilde{O}) = \sqrt{\sum_{i=1}^{n} (\mu_{\tilde{A}}(x_i) - \mu_{\tilde{O}}(x_i))^2}$$

O valor resultante ficara no seguinte domínio

$$0 \leq e(\tilde{A},\tilde{O}) \leq \sqrt{n}$$

Distância de Hamming relativa: É dada pela seguinte fórmula

$$\delta(\tilde{A},\tilde{O})=d(\tilde{A},\tilde{O}) \;/\; n = \; 1/n \sum_{i=1}^{n} \; | \; \mu_{\tilde{A}}(x_i) - \mu_{\tilde{O}}(x_i)|$$

Acima vimos que os valores da distância de Hamming estão dentro do domínio $0 \leq d(\tilde{A},\tilde{O}) \leq n$, dividindo por n obtemos o domínio da distância de Hamming relativa $0 \leq \delta(\tilde{A},\tilde{O}) \leq 1$.

Distância Euclidiana Relativa : O cálculo é dado por

$$e(\tilde{A},\tilde{O})=e(\tilde{A},\tilde{O})/\sqrt{n} \; = \sqrt{1/n \;\; \sum_{i=1}^{n} \; (\mu_{\tilde{A}}(x_i),\mu_{\tilde{O}}(x_i))^2}$$

Pode-se verificar que o cálculo da distância Euclidiana relativa é igual a distância Euclidiana dividida por $\sqrt{n}$ .Desta forma o domínio verdade é dado pelo intervalo $0 \leq \varepsilon \leq 1$.

Esses processos de cálculo de distância possuem vantagens e desvantagens que tornam-se evidentes nas aplicações. Obviamente pode-se definir outras funções além das definidas.

As definições acimas foram baseadas em conjuntos nebulosos finitos (discretos). As operações definidas acima podem ser aplicadas para conjuntos continuos. Entretanto deve-se salientar que quando aplicamos as operações de distância entre conjuntos necessita-se que a somatória seja convergente.

Seja E um conjunto infinito então

$$d(\tilde{A},\tilde{O})= \sum_{i=1}^{\infty} \; | \; \mu_{\tilde{A}}(x_i) - \mu_{\tilde{O}}(x_i)| \quad \text{se a série é convegente.}$$

Similarmente temos $e(\tilde{A},\tilde{O}) \; = \sqrt{\sum_{i=1}^{\infty} \; (\mu_{\tilde{A}}(x_i) - \; \mu_{\tilde{O}}(x_i))^2}$ se a série é convergente.

Conjunto ordinário mais próximo de um conjunto nebuloso: Um conjunto ordinário que, em relação a um dado conjunto nebuloso $\tilde{A}$, possui a menor distância Euclidiana é chamado de conjunto ordinário mais próximo ao conjunto nebuloso $\tilde{A}$. É dado por:

$$\begin{aligned} \mu_A(x_i) = \;& 0 \text{ se } \mu_{\tilde{A}}(x_i) < 0.5 \\ = \;& 1 \text{ se } \mu_{\tilde{A}}(x_i) > 0.5 \\ = \;& 0 \text{ ou } 1 \text{ se } \mu_{\tilde{A}}(x_i)=0.5 \end{aligned}$$

por convenção tomamos, para $\mu_{\tilde{A}}(x_i)=0.5$, o valor de $\mu_A(x_i)=0$.

Índice de nebulosidade: Há duas formas de cálculo do índice de nebulosidade. Uma definida através da distância de Hamming e outra definida através da distância Euclidiana relativa. O cálculo é efetuado utilizando conjunto nebuloso ($\tilde{A}$) e seu conjunto ordinário mais próximo (A). As fórmulas são

$$\nu(\tilde{A}) = 2/n \; d(\tilde{A},A),$$

$$\eta(\tilde{A}) = 2/\sqrt{n} \; e(\tilde{A},A).$$

O número 2 aparece multiplicando porque

$$0 \leq \delta(\tilde{A},A) \leq 0.5 \quad e \quad 0 \leq \varepsilon(\tilde{A},A) \leq 0.5, \; e$$

obtemos o seguinte domínio

$$0 \leq \eta(\tilde{A},A) \leq 1 \qquad 0 \leq \nu(\tilde{A},A) \leq 1$$

Podemos utilizar conjuntos continuos, entretanto a somatória devem ser convergente.

Subconjunto ordinário de nível α : Seja $\alpha \in [0.1]$, chamaremos de conjunto ordinário de nível α, de um subconjunto nebuloso, o subconjunto ordinário dado por

$$A_\alpha = \{ x \mid \mu_{\tilde{A}}(x) \geq \alpha \}.$$

Propriedade importante é

$$\alpha_1 \geq \alpha_2 \Rightarrow A_{\alpha_1} \subset A_{\alpha_2}$$

Teorema da decomposição: Os subconjuntos nebulosos podem ser decompostos como produto dos coeficientes α_i pelo subconjunto ordinário A_{α_i}:

$$\tilde{A} = MAX_{\alpha_i}(\alpha_1.A_{\alpha_1}, \; \alpha_2.A_{\alpha_2}, \ldots, \alpha_n.A_{\alpha_n}), \; 0 \leq \alpha_i \leq 1,$$

$i = 1, 2, \ldots, n.$

Este teorema é facilmente provado tomando

$$\mu_{\tilde{A}_{\alpha_i}}(x) = 1 \qquad \text{se } \mu_{\tilde{A}}(x) \geq \alpha_i$$

$$= 0 \qquad \text{se } \mu_{\tilde{A}}(x) < \alpha_i$$

2.2.2 PROPRIEDADES DOS CONJUNTOS NEBULOSOS

Dado Ã, Õ e Ñ subconjuntos nebulosos de E. As propriedades abaixo são satisfeitas para todos os conjuntos nebulosos. Deve-se salientar que, excetuando as operações de complemento de conjuntos ordinários $A \cup A_c = E$ e $A \cap A_c = \phi$, a álgebra nebulosa obedece as mesmas propriedades da álgebra ordinária.

$\tilde{A} \wedge \tilde{O} = \tilde{O} \wedge \tilde{A}$

$\tilde{A} \vee \tilde{O} = \tilde{O} \vee \tilde{A}$ comutativa

$(\tilde{A} \wedge \tilde{O}) \wedge \tilde{N} = \tilde{A} \wedge (\tilde{O} \wedge \tilde{N})$

$(\tilde{A} \vee \tilde{O}) \vee \tilde{N} = \tilde{A} \vee (\tilde{O} \vee \tilde{N})$ associativa

$\tilde{A} \wedge \tilde{A} = \tilde{A}$

$\tilde{A} \vee \tilde{A} = \tilde{A}$ identidade

$\tilde{A} \wedge (\tilde{O} \vee \tilde{N}) = (\tilde{A} \wedge \tilde{O}) \vee (\tilde{A} \wedge \tilde{N})$

$\tilde{A} \vee (\tilde{O} \wedge \tilde{N}) = (\tilde{A} \vee \tilde{O}) \wedge (\tilde{A} \vee \tilde{N})$ distributiva

$\tilde{A} \wedge \phi = \phi$, onde ϕ é um conjunto ordinário tal que $\forall x_i \in E: \mu_\phi(x_i) = 0$,

$\tilde{A} \vee \phi = \tilde{A}$

$\tilde{A} \wedge E = \tilde{A}$, onde E é um conjunto ordinário tal que $\forall x_i \in E: \mu_E(x_i) = 1$, que é o conjunto de referência.

$\tilde{A} \vee E = E$

$(\tilde{A}_c)_c = \tilde{A}$, involução

$(\tilde{A} \wedge \tilde{O})_c = \tilde{A}_c \vee \tilde{O}_c$

$(\tilde{A} \vee \tilde{O})_c = \tilde{A}_c \wedge \tilde{O}_c$ teorema de Morgan

Produto e soma algébrica de dois conjuntos nebulosos: Seja E um conjunto e M= [0,1] o conjunto de pertinência associado, seja Ã e Õ conjuntos nebulosos de E. Define-se produto algébrico de Ã e Õ, ao qual denota-se por Ã.Õ, como

$$\forall x \in E : \mu_{\tilde{A}.\tilde{O}}(x) = \mu_{\tilde{A}}(x).\mu_{\tilde{O}}(x).$$

A soma algébrica dos dois conjuntos, denotada por Ã ∓ Õ, é dada por:

$$\forall x \in E : \mu_{\tilde{A}\mp\tilde{O}}(x) = \mu_{\tilde{A}}(x)+\mu_{\tilde{O}}(x)-\mu_{\tilde{A}}(x).\mu_{\tilde{O}}(x).$$

Considerando as operações "." e "∓" sobre conjuntos nebulosos, pode-se verificar, abaixo, que as propriedades desses operadores são mais restritas que as propriedades efetuadas pelos operadores "∧" e "∨".

$\tilde{A}.\tilde{O} = \tilde{O}.\tilde{A}$

$\tilde{A} \mp \tilde{O} = \tilde{O} \mp \tilde{A}$ comutativa

$(\tilde{A}.\tilde{O}).\tilde{N} = \tilde{A}.(\tilde{O}.\tilde{N})$

$(\tilde{A} \mp \tilde{O}) \mp \tilde{N} = \tilde{A} \mp (\tilde{O} \mp \tilde{N})$ associativa

$\tilde{A} .\phi = \phi$

$\tilde{A} \mp \phi = \tilde{A}$

$\tilde{A} . E = \tilde{A}$

$\tilde{A} \mp E = E$

$(\tilde{A}_c)_c = \tilde{A}$ involução

$(\tilde{A} . \tilde{O})_c = \tilde{A}_c \mp \tilde{O}_c$

$(\tilde{A} \mp \tilde{O})_c = \tilde{A}_c . \tilde{O}_c$ teorema de Morgan

Pode-se verificar que as propriedades identidade, distributiva e complemento não são satisfeitas. As operações "∧" e "∨" não são distributivas em relação aos operadores "." e "∓".

Deve-se notar que os operadores ".","∓","∧" e "∨" não são aditivos (incrementam a nebulosidade) e portanto não incrementam a nebulosidade de um subconjunto nebuloso (Ã) em operações feitas no universo do discurso E.

Os operadores "." e "∓" são também conhecidos como operadores nebulosos probabilísticos.

As definições efetuadas acima levaram em consideração conjuntos discretos. As operações podem ser aplicadas em conjuntos continuos.

2.3 GRAFOS NEBULOSOS

Considere dois conjuntos E_1 e E_2, seja x um elemento de E_1 e y um elemento de E_2. O conjunto de pares (x,y) define o produto dos conjuntos $E_1 \times E_2$.

Denominamos grafo nebuloso o subconjunto nebuloso $\tilde{N}$ tal que para:

$$\forall \ (x,y) \in E_1 \times E_2: \ \mu_{\tilde{N}}(x,y) \in M,$$

onde M é o conjunto de pertinência do conjunto $E_1 \times E_2$.

Exemplificando temos os conjuntos

$$E_1=\{x_1,x_2,x_3\}, \qquad E_2=\{y_1,y_2\} \text{ e } \qquad M=[0,1]$$

$$E_1 \times E_2=\{(x_1,y_1),(x_1,y_2),(x_2,y_1),(x_2,y_2),(x_3,y_1),(x_3,y_2)\}$$

Sendo $\mu(x_i,y_j) = \mu_{\tilde{N}}(x_i,y_j)$ a função de pertinência. A representação de um grafo nebuloso é dado por:

$$\tilde{N} = \{(x_1,y_1)|0.3, \ (x_1,y_2)|0.7, \ (x_2,y_1)|1.0, \ (x_2,y_2)|0.0, \ (x_3,y_1)|0.5, \ (x_3,y_2)|0.9\}$$

A generalização de grafo nebuloso, para n conjuntos $E_1 \times E_2 \times .. \times E_n$, é um subconjunto nebuloso tal que $x_i \in E_i$

$$i = 1,2,..,n; \ \forall \ (x_1,x_2,..,x_n) \in E_1 \times E_2 \times .. \times E_n;$$

$$\mu(x_1,x_2,..,x_n) \in M, \text{ onde M é o conjunto de pertinência.}$$

2.4 RELAÇÕES NEBULOSAS

A noção de grafo nebuloso pode ser explanada em termos de uma relação nebulosa. Seja P um conjunto de n conjuntos e M o conjunto da sua função de pertinência. Uma relação nebulosa de n argumentos é um subconjunto de P tomando valores em M.

Uma relação nebulosa *R* em $E_1 \times E_2$ ou de E_1 para E_2 é caracterizada como uma função de pertinência

$$\mu_R : E_1 \times E_2 \rightarrow [0,1]$$

onde $E_1 \times E_2$ é o produto cartesiano de E_1 e E_2. Uma relação nebulosa *R* é geralmente descrita na literatura especializada da seguinte forma

$$R = \{ \mu_R(x_1,y_1)/<x_1,y_1>, \ \mu_R(x_1,y_2)/<x_1,y_2>,...,\mu_R(x_n,y_m)/<x_n,y_m>\}$$
$$= \mathbb{R}_{i,j} \ \mu_R(x_n,y_m)/<x_n,y_m>\}$$

onde x_i, i=1,2,..,n $\in E_1$ e y_j, j=1,2,..,m $\in E_2$; e $<x_i,y_j>$ são elementos de *R* . Denotaremos uma relação nebulosa por letras itálicas maiúsculas.

Exemplo: Na tabela abaixo exemplificamos uma relação dada por:

$E_1=\{x_1,x_2,x_3\}$ $E_2=\{y_1,y_2,y_3,y_4\}$ $M=[0,1]$

R	y_1	y_2	y_3	y_4
x_1	0.4	0.9	0.7	1.0
x_2	0.6	0.6	0.2	0.5
x_3	0.3	0.1	0.8	0.4

Nas próximas definições utilizaremos o símbolo

$\wedge_x$ representando o mínimo com respeito a um elemento ou variável x e

$\vee_x$ representando o máximo com respeito a um elemento ou variável x.

Portanto temos $\mu_1(x)= \vee_y \mu(x,y)$, equivalente a $\mu_1(x)= MAX_y \mu(x,y)$, representando o maior valor no sentido x e $\mu(y)= \wedge_x \mu(x,y)$, equivalente a $\mu_1(y)= MIN_x \mu(x,y)$, representando o menor valor no sentido y.

Projeção nebulosa : Dado uma relação nebulosa R, a função de pertinência $\mu_{R_1}(x) = \vee_y \mu_R(x,y)$ define a primeira projeção de R (projeção em relação a x). Da mesma forma a função de pertinência $\mu_{R_2}(y) = \vee_x \mu_R(x,y)$ define a segunda projeção de R (projeção em relação a y).

Aplicando a segunda projeção na primeira projeção (e vice-versa) obtemos a projeção global de uma relação nebulosa, a qual denotamos por $h(R)$. A função é dada por

$$h(R) = \vee_x \wedge_y \mu_R(x,y)$$

$$= \vee_y \wedge_x \mu_R(x,y).$$

Se $h(R)=1$, a relação é denominada normal, se menor que 1 é denominada de subnormal.

Suporte de uma relação nebulosa: Chama-se suporte de uma relação R o conjunto ordinário de pares ordenados (x,y) para os quais a função de pertinência é diferente de zero

$$S(R) = \{(x,y) \mid \mu_R(x,y) > 0\}$$

Envelope de uma relação nebulosa : Sejam R e L duas relações nebulosas tais que

$$\forall (x,y) \in E_1 \times E_2: \mu_R(x,y) \leq \mu_L(x,y),$$

então diz-se que L é um envelope de R ou R é coberto por L.

União de duas relações: A união de duas relações R e L, denotada por $R \vee L$, ou $R + L$, é uma relação tal que

$$\mu_{R \vee L}(x,y) = \mu_R(x,y) \vee \mu_L(x,y)$$
$$= \text{MAX}\,[\mu_R(x,y) , \mu_L(x,y)]$$

Se $R_1, R_2, .., R_n$ são relações então

$$\mu_{R_1, R_2, .., R_n}(x,y) = \vee_{R_i} \mu_{R_i}(x,y).$$

Interseção de duas relações: A interseção de duas relações R e L, denotada por $R \wedge L$, é uma relação tal que

$$\mu_{R \wedge L}(x,y) = \mu_R(x,y) \wedge \mu_L(x,y)$$
$$= \text{MIN}\,[\mu_R(x,y) , \mu_L(x,y)]$$

Generalizando para n relações seja $R_1, R_2, .., R_n$ então

$$\mu_{R_1, R_2, .., R_n}(x,y) = \wedge_{R_i} \mu_{R_i}(x,y).$$

Produto algébrico entre relações: Definimos o produto algébrico entre duas relações R e L, denotada por $R \,.\, L$, pela operação definida abaixo:

$$\mu_{R.L}(x,y) = \mu_R(x,y) \,.\, \mu_L(x,y).$$

Distributividade: Dado três relações nebulosas R, L, e Q temos

$$R \wedge (L \vee Q) = (R \wedge L) \vee (R \wedge Q)$$
$$R \vee (L \wedge Q) = (R \vee L) \wedge (R \vee Q)$$
$$R \,.\, (L \vee Q) = (R \,.\, L) \vee (R \,.\, Q)$$
$$R \,.\, (L \wedge Q) = (R \,.\, L) \wedge (R \,.\, Q)$$

Soma algébrica de relacões nebulosas: Definimos a soma algébrica entre duas relações R e L, denotada por $R \mp L$, pela operação definida abaixo:

$$\mu_{R \mp L}(x,y) = \mu_R(x,y) + \mu_L(x,y) - \mu_R(x,y) \,.\, \mu_L(x,y)$$

Notam-se duas propriedades distributiva: Dadas as seguintes relações nebulosas R, L e Q temos

$$R \mp (L \vee Q) = (R \mp L) \vee (R \mp Q)$$
$$R \mp (L \wedge Q) = (R \mp L) \wedge (R \mp Q)$$

Complemento de uma relação: O complemento de uma relação R, denotada por R_c, é dada por uma relação tal que

$$\forall\, (x,y) \in E_1 \times E_2 : \mu_{R_c}(x,y) = 1 - \mu_R(x,y).$$

Soma disjuntiva de duas relações: A soma disjuntiva de duas relações nebulosas R e L, denotada por $\oplus$, é dado por

$$R \oplus L = (R \wedge L) \vee (R \wedge L).$$

Relação ordinária fechada para uma relação nebulosa: Da mesma forma que definimos conjunto ordinário mais próximo, definimos uma relação K fechada para R que é dada por

$$\mu_K(x,y) = 0 \text{ se } \mu_R(x,y) < 0.5$$
$$= 1 \text{ se } \mu_R(x,y) > 0.5$$
$$= 0 \text{ ou } 1 \text{ se } \mu_R(x,y) = 0.5 .$$

Por convenção tomamos, para $\mu_R(x,y)=0.5$ o valor de $\mu_K(x,y)=0$.

Esta definição é útil quando referenciamos conjuntos de E_1 e E_2, dado por $E_1 \times E_2$, onde $x \in E_1$ e $y \in E_2$, e consideramos os conjuntos finitos ou não.

2.4.1 COMPOSIÇÃO DE RELAÇÕES NEBULOSAS

Composição max-min: Seja $R_1 \subset X \times Y$ e $R_2 \subset Y \times Z$. Definimos a composição max-minimo de R_1 e R_2, denotada por $R_2 \circ R_1$, pela expressão

$$\mu_{R_1 \circ R_2}(x,z) = \vee_y [\mu_{R_1}(x,y) \wedge \mu_{R_2}(y,z)]$$
$$= \text{MAX } [\text{MIN } [\mu_{R_1}(x,y) , \mu_{R_2}(y,z)]]$$

onde $x \in X$, $y \in Y$ e $z \in Z$.

Composição max-min é associativa e a seguinte afirmação é válida

$$(R_3 \circ R_2) \circ R_1 = R_3 \circ (R_2 \circ R_1).$$

Se R é uma relação definida sobre $E \times E$, então $R \subset E \times E$, podemos escrever

$$R \circ R = R^2$$
$$R \circ R^2 = R^2 \circ R = R^3 \text{ e generalizando para k termos temos}$$
$$R \circ R \circ R \ldots R = R^k.$$

A propriedade distributiva da composição max-min é aplicável com a operação união, mas não com a operação interseção $R \circ (L \vee Q) = (R \circ L) \vee (R \circ Q)$

$$R \circ (L \wedge Q) \neq (R \circ L) \wedge (R \circ Q)$$

Outra importante propriedade é dada a seguir

$$L \subset Q \Rightarrow R \circ L \subset R \circ Q$$

Composição max-operação : Podemos substituir a operação $\wedge$ (min) da definição max-min, por alguma outra operação contanto que seja associativa e não monotônica crescente em cada argumento. O simbolo * denota esta operação.

$$\mu_{L*R}(x,z) = \vee_y [\mu_R(x,y) * \mu_L(y,z)]$$

Composição max-produto : Entre os operadores possíveis de serem utilizados na max-operação está o operador '.'(produto), pois obedece às restrições. Portanto a fórmula anterior fica.

$$\mu_{L.R}(x,z) = \vee_y [\ \mu_R(x,y) \ . \ \mu_L(y,z)\ ] \ .$$

Subconjunto nebuloso de nível α de uma relação nebulosa : Seja $\alpha \in [0,1]$, chamaremos de conjunto ordinário de nível α de uma relação nebulosa $R \subset X \times Y$, o subconjunto ordinário dado por:

$$G_\alpha = \{\ (x,y) \mid \mu_R(x,y) \geq \alpha\ \}$$

Teorema da decomposição : Uma relação nebulosa R pode ser decomposta na forma:

$$R = \vee_\alpha \alpha\ R_\alpha, \quad 0 \leq \alpha \leq 1$$

onde $\quad \mu_{R_\alpha}(x,y) = 1 \quad$ se $\quad \mu_R(x,y) \geq \alpha$

$$\mu_{R_\alpha}(x,y) = 0 \quad \text{se} \quad \mu_R(x,y) < \alpha$$

A notação $\alpha\ R_\alpha$ indica que todos os elementos da relação ordinária R_α são multiplicados por α. Para recuperarmos o conjunto R basta aplicarmos a operação união a todos os $\alpha\ R_\alpha$ gerados na decomposição.

Subconjunto nebuloso condicionado : Sejam os subconjuntos nebulosos $\tilde{A} \subset E_1$ e $\tilde{N} \subset E_2$, e a relação nebulosa R. O subconjunto nebuloso $\tilde{A}$ induz o subconjunto nebuloso $\tilde{N}$ através da relação R. Pode-se também obter $\tilde{A}$ a partir de $\tilde{N}$ e R.

$$\tilde{A} \xrightarrow[R]{} \tilde{N}$$

Se $\mu_R(x,y)$ é a função de pertinência da relação R, e as funções de pertinência $\mu_{\tilde{A}}(x)$ e $\mu_{\tilde{N}}(y)$ de $\tilde{A}$ e $\tilde{N}$ respectivamente, podemos deduzir a seguinte equação

$$\mu_{\tilde{N}}(y) = \text{MAX}_{x \in E_1} \text{MIN}\ [\mu_{\tilde{A}}(x),\ \mu_R(x,y)]$$

$$= \vee_x [\ \mu_{\tilde{A}}(x) \wedge \mu_R(x,y)]$$

Esta operação é uma das principais regras de inferência, usada para operar sobre dados nebulosos.

2.4.2 PROPRIEDADES DAS RELAÇÕES NEBULOSAS

Simétrica : Uma relação nebulosa é simétrica se

$$\forall\ (x,y) \in E \times E : (\mu_R(x,y) = \mu\ ) \Rightarrow (\mu_R(y,x) = \mu\ ).$$

Reflexiva : Uma relação nebulosa é reflexiva se

$$\forall\ (x,x) \in E \times E : \mu_R(x,x) = 1.$$

Transitiva : Uma relação nebulosa é transitiva se

$$\forall (x,y), (y,z), (x,z) \in E_1 \times E_2$$

$$\mu_R(x,z) \geq MAX_Y [MIN(\mu_R(x,y),\mu_R(y,z))].$$

Pré-ordem : Uma relação nebulosa é uma relação em pré-ordem se as seguintes condições forem obedecidas:

a) reflexiva

b) transitiva

Equivalência : Uma relação nebulosa é uma relação de equivalência se as sequintes propriedades forem obedecidas:

a) reflexiva

b) transitiva

c) simétrica

Semelhança : Uma relação nebulosa é uma relação de semelhança se as sequintes propriedades forem obedecidas:

a) reflexiva

b) transitiva

2.5 VARIÁVEIS LINGUÍSTICAS

Variáveis linguísticas são palavras ou sentenças na linguagem natural ou sintética. Quando usamos o termo "idade" intuitivamente associamos a ele conjuntos nebulosos tais como "jovem", "não jovem", "muito jovem", "velho", etc..

Os conjuntos nebulosos atribuídos podem ser discretos ou contínuos. Para conjuntos contínuos escolhe-se uma função matemática que descreva o comportamento da variável. As funções matemáticas traçadas no gráfico abaixo representam o comportamento dos conjuntos nebulosos "jovem" e "não jovem". Portanto o usuário escolhe num conjunto de funções matemáticas nebulosas préviamente definidas, àquela que melhor adapta-se a variável linguística que será utilizada.

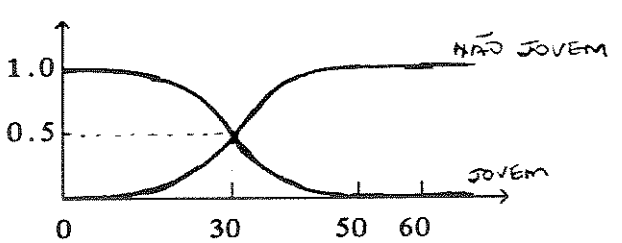


Fig 2.1 : Gráfico das funções jovem e não jovem

Em algumas aplicações, o conjunto nebuloso fornecido é discreto, e não adapta-se a nenhuma função matemática nebulosa conhecida. Abaixo define-se um conjunto nebuloso discreto.

febre = { 0.2 /33, 0.5/37, 0.8/38, 1.0/39 }

Quando trabalhamos com conjuntos nebulosos discretos podemos necessitar de valores intermediários aos contidos no conjunto. Por exemplo, o valor para febre igual a 37,5 graus no conjunto acima. A solução é a utilização de métodos de interpolação. Utilizando os valores discretos contidos no conjunto geramos através de interpolação, uma função aproximada ou um valor intermediário. Geralmente os valores são obtidos através de interpolação linear. Então utilizando interpolação linear obtemos, para febre igual a 37,5 graus, o valor 0.65.

Geralmente os valores das variáveis linguísticas são obtidos por conjuntos nebulosos primários (por exemplo "jovem" ou por seu antônimo "velho"), ou por uma coleção de modificadores ("não", "muito", "mais ou menos", "inteiramente", "não muito", "aproximadamente",etc..) que operam sobre os conjuntos nebulosos primários, e por conectivos "e" e "ou" que conectam conjuntos nebulosos primários ou conjuntos modificados. Assim utilizando as definições acima, o valor da variável linguística "idade" pode ser dada por "não muito jovem e não muito velho".

A implementação do modificador "muito" é efetuada pela operação de concentração, denotado por CON(Ã) e definida por :

$$CON(\tilde{A}) = \tilde{A}^2 \text{ onde } \tilde{A} \text{ é um conjunto nebuloso.}$$

Da mesma forma o modificador "aproximadamente" é implementado pela operação de dilatação, denotada por DIL(Ã) e expresso por

$$DIL(\tilde{A}) = \tilde{A}^{0.5} \text{ sendo } \tilde{A} \text{ um conjunto nebuloso.}$$

Assim como definimos os operadores "muito" e "aproximadamente", pode-se definir as outras operações. Para melhor compreensão de operadores, imagine que João possua o grau de pertinência 0.8 no conjunto nebuloso "jovem". Então ele terá grau de pertinência 0.64 (0.8^2) no conjunto nebuloso "muito jovem" e grau 0.89 no conjunto "aproximadamente jovem".

Deve-se frisar que manipulamos os conjuntos nebulosos, que representam variáveis linguísticas, de maneira intuitiva. Trabalhos têm demonstrado que todos os seres humanos utilizam funções e operadores semelhantes às definidas neste trabalho, para descrever as variáveis linguísticas. Foi verificado que os operadores modificadores são executados intuitivamente pelas pessoas. Por exemplo o operador "muito" é equivale a aplicação da concentração. Um fato a salientar é que todos os trabalhos nesta área foram efetuados com pessoas de países de língua inglesa, e não existem estudos referentes à manipulação de variáveis linguísticas na língua

portuguesa.

2.6 EXTENSÕES DE CONJUNTOS E RELAÇÕES NEBULOSAS

Extensões de conjuntos e relações nebulosas são estruturas que permitem a definição de tipos de dados nebulosos mais complexos. Abaixo definiremos três novos tipos de dados nebulosos. Seja E o universo do discurso.

L-conjunto nebuloso : Um L-conjunto nebuloso em E é caracterizado pela seguinte função de pertinência

$\mu(x) : E \rightarrow L$ onde L representa um reticulado .

Trata-se de uma generalização do espaço de pertinência do intervalo [0,1] para o reticulado L.

Abaixo vemos um exemplo de um L-conjunto nebuloso.

Dado E={ a,b,c,d}, temos

$$X = \{<0.1,0.7>/a, \quad <0.5,0.2>/b, \quad <0.3,0.9>/c\}$$

como um L-conjunto nebuloso em E.

Conjunto nebuloso de nível-m : Este tipo de dado nebuloso é semelhante aos conjuntos nebulosos. Entretanto os elementos deste conjunto são conjuntos nebulosos. O conjunto nebuloso de nível-m em E é caracterizado pela seguinte função de pertinência

$$\mu(y) : [0,1]|\ [0,1]|\ \ldots|[0,1]^E \rightarrow [0,1]$$

onde $A^B \equiv A \mid B$ são todas as funções de B para A . O universo do discurso de um conjunto nebuloso de nível-m deve ser um conjunto nebuloso de nível-(m-1). Para conjuntos nebulosos de nivel-1 o universo é um conjunto ordinário.

Como exemplo temos: seja E={a,b,c,d} e dois conjuntos nebulosos em E Y1={0.3/a, 0.7/c, 0.4/d} e Y2={0.5/b, 0.8/d} podemos definir um conjunto nebuloso nível-2 por Y={0.7/Y1, 0.1/Y2}.

Conjunto nebuloso tipo-n : Este tipo de dado nebuloso é semelhante ao conjunto nebuloso, entretanto o grau de pertinência é dado por um conjunto de números nebulosos. Uma conjunto nebuloso de tipo-n em E é definido por uma função de pertinência dada por:

$$\mu(z) : E \rightarrow [0,1]|\ [0,1]|\ \ldots|\ [0,1]$$

O valor do grau de pertinência são conjuntos nebulosos tipo-(n-1) do intervalo [0,1] ao invés de pontos em [0,1]. Conjuntos nebulosos do tipo-1 são equivalentes a conjuntos nebulosos ordinários. Seja E definido anteriormente, podemos definir um conjunto nebuloso tipo-2

$$Z = \{\textbf{alto}/a\ ,\ \textbf{médio}/b,\ \textbf{baixo}/c\}$$

onde **alto**, **médio** e **baixo** são conjuntos nebulosos em {0, 0.1, ..., 1} $\subseteq$ [0,1].

2.7 LÓGICA NEBULOSA

Lógica, de acordo com o dicionário, é a ciência que estuda as leis do raciocínio. A lógica clássica nos possibilita representar e inferir fatos de informações precisas.

Entretanto sabemos que frequentemente raciocinamos em ambientes imprecisos e incertos, e a lógica clássica não nos permite manipular este tipo de conhecimento, pois não possui formalismo para representar fatos e informações imprecisas, e não possui mecanismos para inferência com dados imprecisos.

A lógica nebulosa tem sido proposta como um formalismo que, além da lógica clássica, possibilita a modelagem dos modos imprecisos de raciocínio e a criação de mecanismos para efetuar inferência com dados parciais e imprecisos. Dentro desse conceito, formalizaremos a seguir propostas de operadores nebulosos e mecanismos de inferência nebulosos originados da lógica simbólica e probabilidade. Destacaremos na próxima seção a proposta de um modelo onde a representação das informações é baseada em grafos nebulosos e um único operador nebuloso.

A lógica nebulosa foi criada com base no conceito dos conjuntos nebulosos e da lógica simbólica. Na lógica simbólica manipulamos valores "verdade" (1 se o fato está inserido no conjunto do discurso) e "falso" (0 se o fato não está inserido no conjunto do discurso). Na lógica nebulosa esse conceito foi extendido e os valores manipulados estão contidos no intevalo [0,1], portanto podemos manipular valores intermediários (0.7, 0.3, etc). Na lógica simbólica os operadores básicos utilizados são "e" e "ou". Na lógica nebulosa foram inicialmente propostos os operadores nebulosos: "e" (denotado por "&" que consiste na aplicação da operação mínimo definida na teoria dos conjuntos nebulosos) e o operador nebuloso "ou" (denotado por $\vee$ que consiste na aplicação da operação máximo, também definida na teoria dos conjuntos nebulosos).

Portanto podemos formalizar a lógica nebulosa por

$$(U, \{\neg, \&, \vee\}, [0,1])$$

onde : U é o conjunto do universo do discurso;

$\{\neg, \&, \vee\}$ são os operadores ou conectores básicos e equivalente as operações definidas para conjuntos nebulosos

"$\neg$" - complemento, "&" - mínimo, "$\vee$" - máximo;

[0,1] é o domínio dos valores verdade.

Uma variável será denotada por literais ("x", "y", etc) e sua negação por com $\neg x$. O complemento é dado por $\neg x = 1 - x$.

Uma fórmula nebulosa,composta de variáveis nebulosa x_1, $x_2 \ldots x_n$, e conectadas por operadores nebulosos, é denotada por $F(x_1, x_2, \ldots, x_n)$ e representa o mapeamento:

$$F : [0,1]^n \rightarrow [0,1].$$

A lógica nebulosa obedece a todos os axiomas definidos para a lógica Booleana exceto o axioma da complementaridade. Seja F uma fórmula nebulosa

$$F \,\&\, \neg(F) \neq 0 \quad \text{e} \quad F \vee \neg(F) \neq 1.$$

Ao efetuarmos a substituição das variáveis de uma fórmula obtemos seu valor verdade. Chamaremos de T(F) o valor verdade de uma fórmula nebulosa F, e $\mathbb{F}$ o conjunto das fórmulas nebulosas.

O procedimento de avaliação dos valores verdades de uma fórmula nebulosa pode ser descrito por:

$T(F) = T(A)$ se $F=A$ e A uma fórmula atômica.

$T(F) = 1-T(R)$ se $F= \neg R$.

$T(F) = \min[T(F_1), T(F_2)]$ se $F=F_1 \,\&\, F_2$.

$T(F) = \max[T(F_1),T(F_2)]$ se $F = F_1 \vee F_2$.

$T(F) = \inf[T(B(x)) \mid x \in U]$ se $F = (x)B$ e U é o domínio de x.

$T(F) = \sup[T(B(x)) \mid x \in U]$ se $F = (Ex)B$ e U é o domínio de x.

Se U é um conjunto finito, então as duas últimas regras tornan-se:

$T(F) = T(B(a_1) \,\&\, \ldots \,\&\, B(a_n))$ se $F = (x)B$ e x assume valores em $a_1, \ldots, a_n$.

$T(F) = T(B(a_1) \vee \ldots \vee B(a_n))$ se $F = (Ex)B$ e x assume valores em $a_1, \ldots, a_n$.

Na lógica clássica temos que uma fórmula F é verdadeira se T(F) = 1. Para a lógica nebulosa podemos afirmar que uma fórmula nebulosa A é verdadeira se $T(A) \geq 0.5$, como consequência $T(A) \geq T(\neg A)$.

Definição. 1: Uma fórmula nebulosa $F \in \mathbb{F}$ é dita ser válida (inconsistente) se $T(F) \geq 0.5$ ($T(F) \leq 0.5$) para todas as atribuições das variáveis em F.

Uma fórmula $F \in \mathbb{F}$ é dita ser inválida (consistente) se F não é válida (não inconsistente).

Uma cláusula nebulosa é uma disjunção de literais (L1 $\vee$ L2 $\vee$ ...$\vee$ Ln); e uma frase nebulosa é uma conjunção de literais (L1 & L2 & ...& Ln).

Definição 2: Uma fórmula nebulosa F é uma forma normal disjuntiva (conjuntiva) se

$F = \phi_1 \vee \phi_2 \vee \ldots \phi_p$, $p \geq 1$ onde ϕ_j são frases ($F = C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_p$, $p \geq 1$, onde C_j são cláusulas).

Estas duas formas serão denotadas por DNF e CNF respectivamente.

Teorema 1 : Uma fórmula nebulosa F $\in$ F é uma fórmula nebulosa válida (fórmula nebulosa inconsistente) se e somente se F é válida (inconsistente).

Para provar necessitamos dos seguintes lemas:

Lema 1 : a) Uma cláusula C é uma nebulosa válida se e sómente se contém um par de variáveis $(x_i, \neg x_i)$;

b) Uma frase ϕ é uma nebulosa inconsistente se e somente se contém um par de variáveis $(x_i, \neg x_i)$.

Prova :

a) Se C contém o par complementar $(x_i, \neg x_i)$, então de C= L1 $\vee$ L2 ...Ln temos:

$$T(C) = \max_{1 \le j \le m} T(L_j) \ge T(x_i, \neg x_i) \ge 0.5$$

Portanto C é uma cláusula nebulosa válida.

Por outro lado, se C é válida, e não possui o par complementar $(x_i, \neg x_i)$. Então sendo C = L1 $\vee$ L2 $\vee$ ...$\vee$ Ln, podemos supor uma atribuição tal que

$$T(Li) < 0.5 \ , \quad \text{para } 1 \le i \le n.$$

Então T(C) $\le$ 0.5 e C não é válida, portanto a) é provado por contradição.

b) Seja ϕ = L1 & L2 & ... & Ln uma frase nebulosa; se ϕ contém um par $(x_i, \neg x_i)$ então:

$$T(\phi) = \min_{1 \le j \le m} T(L_j) \le T(x_i, \neg x_i) \le 0.5.$$

Portanto ϕ é inconsistente.

A volta pode ser provada como a).

Podemos claramente verificar a dualidade entre a) e b).

Lema 2 : Uma formula nebulosa $F=\phi_1 \vee ... \vee \phi_p$, em D.N.F. ($F= C_1 \ \& \ C_2 \ \& ... \& C_p$, C.N.F.) é uma nebulosa inconsistente (nebulosa válida), se e somente se todas $\{\phi_j\}_{j=1,p}$ são nebulosa inconsistente (todos $\{C_j\}_{j=1,p}$ são nebulosas válidas).

Prova : $T(F) = \max T(\phi_j)_{j=1,p}$ e $T(F) \le 0.5 \Leftrightarrow T(\phi_j) \le 0.5$ para todo j=1,..,p, portanto F é nebulosa inconsistente $\Leftrightarrow$ $\{\phi_j\}_{j=1,p}$ são nebulosa inconsistente (todos $\{C_j\}_{j=1,p}$ são nebulosas válidas).

A outra parte é provada por dualidade.

Prova do teorema 1. Suponha F ser uma nebulosa válida. Portanto T(F) $\ge$ 0.5 para todas as atribuições das variáveis em F. Suponha atribuições dois valores {0,1} para as variáveis, neste caso sempre teremos T(F) $\ge$ 0.5 .

Portanto, neste caso, temos somente T(F) $\ge$ 1, portanto F é válida.

Por outro lado, se F é válida (em dois valores), supomos primeiro , que F é uma cláusula

$$F= L_1 \vee L_2 \vee ...\vee L_p$$

Para cada $T(L_j) \in \{0,1\}$, $1 \leq j \leq p$, temos T(F)=1 e F=1. Existe i, j, $k \in \{1,..,p\}$ tal que $L_i=x_k$, $L_j= \neg x_k$ ($x_k \vee \neg x_k = 1$ neste caso). Como F agora contém o par (x_k, $\neg x_k$), seu resultado após o lema 1 é que F é válido.

Do lema 2 resulta que se F está na CNF e é válida, então F é nebulosa válida. A prova é alcançada através da dualidade.

A seguir exemplificamos esses conceitos.

Seja $F= (x_1 \vee x_2) \& x_1 \& (x_2 \vee x_3)$ em CNF.

Se $x_1 \vee x_2$, x_1 e $x_2 \vee x_3$ não são nebulosas válidas, então F não é nebulosa válida.

Pode-se verificar que a definição da lógica nebulosa, utilizando os operadores nebulosos min-max, está próxima dos conceitos da lógica dos predicados e conseqüentemente possibilita implementação de ferramentas do tipo do PROLOG-nebuloso.

Infelizmente os operadores nebulosos max-min não tem se demonstrado suficientemente gerais para modelar o raciocínio humano e outros operadores tem sido propostos para suprir as deficiências.

Existem propostas de operadores baseados na teoria da probabilidade. As propostas baseiam-se no fato de que o raciocínio seja feito utilizando probabilidades. A certeza e a importância de uma informação é obtida de acordo com sua frequência na experiência diária. Então seleciona-se no curso das ações, baseadas nas evidências e na frequência anterior, aquela que parece ser a melhor opção para produzir o resultado desejado.

O operador probabilidade-ou em lógica nebulosa, sera denotado por "p-ou", é definido a seguir.

Dados dois eventos independentes A e B, a probabilidade de um ou outro ocorrer é dada por

$$p\text{-}ou(A,B) = p(A) + p(B) - (p(A)*p(B))$$

O operador probabilidade-e em lógica nebulosa, denotado por "p-e", é definido a seguir.

Dados dois eventos independentes A e B, a probabilidade de um e o outro ocorrer é dada por:

$$p\text{-}e(A,B) = p(A) * p(B).$$

O operador complemento, que será denotado por "p-nao", é definido a seguir: se conhecemos a probabilidade de um evento A ocorrer, então a probabilidade do evento A não ocorrer é dado por:

$$p\text{-nao}(A) = 1 - p(A).$$

Utilizando os operadores probabilísticos podemos formalizar a lógica nebulosa por

$$(U, \{p\text{-e}, p\text{-ou}, p\text{-nao}\}, [0,1])$$

Existem outras propostas de operadores para a lógica nebulosa. Neste trabalho nos retringiremos aos modelos apresentados acima e ao modelo descrito na seção 2.10.

2.8 CONSEQUÊNCIA LÓGICA E O PRINCÍPIO DA RESOLUÇÃO

Na seção anterior vimos que para a definição de lógica necessitamos de um formalismo para representar fatos e informações, e definir mecanismos de inferência.

Nesta seção estenderemos as definições de algumas regras de inferência e do princípio de resolução, da lógica clássica para o contexto da lógica nebulosa.

Uma das principais regras de inferência é o conceito da consequência lógica (Modus Ponens). A partir de F e de $(F \rightarrow G)$ deduzimos G. Outra representação pode ser dada por F e $(F \,\&\, \neg G)$.

Dada uma fórmula F, definimos uma fórmula G como consequência lógica de F se e sómente se $F\&\neg G$ é inválida. Se $(F \,\&\, \neg G)$ é inválida, então $T(F \,\&\, \neg G) \leq 0.5$ para todas as interpretações. Se requerermos $T(F) \geq 0.5$, então $T(G) \geq 0.5$ para todas as interpretações. Isto significa que se o grau de verdade de F ultrapassa 0.5, o grau de verdade de todas as consequências lógicas de F serão menor que 0.5. A definição de consequência em lógica nebulosa é compatível com a lógica dois valores.

Lema : Uma fórmula G é uma consequência lógica de uma fórmula F se e somente se G é uma consequência lógica na lógica dois valores.

A seguir são dadas algumas regras de inferência. Pode-se verificar que as mesmas são variações da regra "modus ponens", consequentemente podem ser estendidos para a lógica nebulosa.

Modus Tolens : Se $\neg G$ e $(\neg F \vee G)$ então $\neg F$.

Silogismo disjuntivo : Se $(A \vee B)$ e $\neg A$ então B.

Silogismo hipotético : Se $(A \rightarrow B)$ e $(B \rightarrow C)$ então $(A \rightarrow C)$.

Dilema construtivo : Se $(A \rightarrow B) \,\&\, (C \rightarrow D)$ e $(A \vee C)$ então $(B \vee D)$.

Dilema destrutivo : Se $(A \rightarrow B) \,\&\, (C \rightarrow D)$ e $\neg B \vee \neg D$ então $\neg A \vee \neg C$.

Na lógica binária há, uma regra de inferência, chamada de princípio de resolução, a qual provou-se ser completa para deduzir consequências lógicas de cláusulas.

Seja um conjunto de cláusulas **S** e uma cláusula C. Uma dedução de C a partir de **S** consiste de uma sequência de cláusulas terminando em C, e geradas aplicando-se repetidamente a regra de resolução. Uma refutação a partir de **S** é a obtenção da cláusula vazia. Portanto, ao efetuarmos uma consulta de uma cláusula nebulosa C ao conjunto **S** necessitamos obter a cláusula vazia, ou seja refutar a cláusula nebulosa C.

Então seja **S** o conjunto de cláusulas. A resolução de **S**, denotada por R(**S**), é um conjunto contendo membros de **S** e todos os resolventes derivados de algum par de cláusulas de **S**. A n-ésima resolução de **S**, denotada por $R^n(\mathbf{S})$, é definida para $n \geq 0$ a seguir:

$$R^0(\mathbf{S}) = \mathbf{S} \qquad \text{e} \qquad R^{n+1}(\mathbf{S}) = R^1(R^n(\mathbf{S})).$$

Tanto em probabilidade quanto na lógica binária temos que se B é uma consequência de A então $P(B) \geq P(A)$ (ou $T(B) \geq T(A)$ na lógica dois valores). Entretanto este fato não é verdade na lógica nebulosa.

Considere : C1 : $\neg A \vee B$, C2 : A. B é o resolvente de C1 e C2.

Seja T(A)=0.3 e T(B)=0.2 temos

$$T(C1 \,\&\, C2) = T((\neg A \vee B) \,\&\, A)$$
$$= \min\,[\max[0.7,0.2],\ 0.3] = \min[0.7,0.3] = 0.3$$

Portanto $T(B) < T(C1 \,\&\, C2)$.

Como consequência a definição do princípio de resolução nebuloso terá que limitar o domínio para o valor verdade.

Teorema : Seja **S** um conjunto de cláusulas nebulosas. Seja C1,C2,...,Cn cláusulas de **S**. Seja $\max[T(C1), T(C2),...,T(Cm)] = b$ e $\min[T(C1),T(C2),...,T(Cm)] = a$. Seja C^n denotando alguma clausula no conjunto de resoluções de **S**, $R^n(S)$. Então para todo $n \geq 0$, $a \geq T(C^n) \geq b$.

Este teorema não será provado. Podemos verificar consequências lógicas obtidas pela repetição do princípio de resolução terá valor pelo menos maior ou igual que "a" e nunca ultrapassará "b".

Dois outros passos são inseridos na regra de resolução, a fim de gerar cláusulas que implique lógicamente seu consequente. O primeiro passo é quando cláusulas possuem literais complementares. Podemos obter instâncias através de substituições e em seguida reduzir a sua consequência. Dado :

chama(a,z)

¬chama(x,y), depende(x,y)

aplicamos a substituição β={x/a, y/z}, e obtemos

¬chama(a,z), depende(a,z)

e em seguida podemos aplicar a resolução e obtermos depende(a,z).

No passo anterior utilizamos a unificação para substituir literais oriundo de cláusulas diferentes. Pode-se também tornar idênticos literais de uma mesma cláusula. Ilustramos este processo abaixo. Dado

¬p(x), ¬p(a)

obtemos ¬p(a) utilizando a substituição {x/a}.

Estes passos de substituíção ou fatoração são enunciados de forma separada, porém devem ser incluídos como parte do processo de resolução.

2.9 TEORIA DA POSSIBILIDADE

Considere uma proposição q (forma canônica):

q ≅ X é um inteiro pequeno,

onde "inteiro pequeno" é um conjunto nebuloso definido em um universo do discurso,

"inteiro pequeno"=1/0, 1/1, 0.9/2, 0.7/3, 0.5/4, 0.2/5.

A interpretação da proposição é dada por: q induz uma distribuição de possibilidade $\prod_x$, onde para cada inteiro X tem-se associado um valor igual ao grau de pertinência no conjunto "inteiro pequeno".

Portanto Poss{ X=0} = Poss { X=1}= 1, Poss{X=2} =0.9, Poss{ X=3} = 0.7, Poss{ X=4} = 0.5, Poss{X=5}=0.3, Poss{ X=u} = 0 para u < 0 ou u > 5.

Definicão : Se X é uma variável que toma valores em E, e F é um subconjunto nebuloso de E caracterizado por uma função de pertinência μ_F, então a proposição

q ≅ X é F ,

induz uma distribuição de possibilidade $\prod_x$ que é igual a F, $\prod_x$= F implicando que Poss{X=u } = $\mu_F(u)$ para todo u ∈ U.

Deve-se destacar que a definição do subconjunto nebuloso F é efetudada de modo subjetivo na natureza e a distribuição de possibilidade depende do subconjunto nebuloso.

Pode-se verificar que a proposição inicial "X é F" traduz-se na atribuição de um conjunto nebuloso F a uma distribuição de possibilidade X,

$$q \cong X \text{ é } F \quad \text{traduz-se em } \prod_x = F.$$

Um importante fato obtido da definição de $\prod_x$ é a implicação que o grau de possibilidade pode ser algum valor no intervalo [0,1]. Desta forma podemos expressar graus intermediários de possibilidade que estão implícitos em proposições comumente encontrada tais como " É pouco provável minha ida ao Nepal", "Há pouca possibilidade do meteorito cair na minha cabeça".

O conceito de medida de possibilidade e necessidade são consêquencia do conceito de possibilidade. Estes conceitos requerem que a distribuição de possibilidade seja normalizada, sendo que ao menos um elemento tenha possibilidade 1.

A medida de possibilidade é dada por

$$\text{Poss } \{ X \text{ é } F\} \cong p(F)$$
$$= \sup_{u \in U} [\mu_F(u) \ \& \ p_X(u)]$$

onde F é um conjunto nebuloso de U, caracterizado por uma função de pertinência μ_F e p_X é a função da distribuição de possibilidade (ou a função da distribuição de possibilidade de $\prod_X$) que é associada com a variável X que toma valores em U.

Dado uma distribuição normalizada $\prod_{AUSENCIA(BOB)} = 0.3/15, 0.5/16, 1/17, 0.5/18$ e um conjunto nebuloso "meio do mês" (denominado de M)

$\mu_M(u) = 0$ para $1 \leq u \leq 10$ ou $20 \leq u \leq 31$

$\mu_M(11)=\mu_M(19)=0.1$, $\mu_M(12)=\mu_M(18)=0.4$, $\mu_M(13)=\mu_M(17)=0.7$,

$\mu_M(14)=\mu_M(16)=0.9$, $\mu_M(15)=1$

A Poss{AUSÊNCIA(BOB) é (no) "meio do mês"} = p(M),

$P_{BOB}(M) = \max(0.3 \ \& \ 1, \ 0.5 \ \& \ 0.9, \ 1 \ \& \ 0.7, \ 0.5 \ \& \ 0.4) = 0.7$

O operador "&" pode ser substituído pelo operador produto e obteremos possibilidades com valores iguais ou menores que os obtidos com o operador "&".

O conceito de medida de necessidade deriva do significado de necessidade. A necessidade de um evento é vista como a impossibilidade do evento oposto. Definimos como:

$$\text{Nec } \{X \text{ é } F\} \cong n(F)$$
$$= \inf_{u \in U} [(1 - p_x(u)) \vee \mu_F(u)]$$

A necessidade de estar ausente no meio do mês é equivalente a dizer, "Não é possível que X não deva estar ausente no meio do mês". O conceito de necessidade é de difícil compreensão pelo fato das pessoas frequentemente manipularem fatos positivos como possibilidade, probabilidade e certeza.

Para uma mesma distribuição o resultado da medida de necessidade deve ser menor que o da medida de possibilidade.

Efetuando o cálculo de medida de necessidade no exemplo anterior temos:

Nec (AUSÊNCIA(BOB) é (no) meio do mes) =

inf((1-0.3)v1, (1-0.5)v 0.9,(1-1)v 0.7, (1-0.5) v 0.4) = 0.5.

2.9.1 REGRAS DE INFERÊNCIA

Uma característica da lógica nebulosa, é que a premissa e a conclusão são geralmente expressas na forma canônica (p ≅ X é A). Esta representação coloca em evidência o fato de que cada premissa é uma restrição sob uma variável e a conclusão é uma restrição induzida, computada através de um processo de propagação de restrições.

As regras de inferência em lógica nebulosa podem ser classificadas em : Regras categóricas, são regras que não possuem quantificadores. Regras de disposição, são regras em que uma ou mais premissas podem conter, explicitamente ou implicitamente o quantificador nebuloso "geralmente". Um exemplo de regra categórica é a regra de inferência conhecida como princípio de herança:

X é A

A ⊂ B temos X é B onde X toma valores no universo do discurso U, A e B são subconjuntos nebulosos de U.

Uma regra de disposicão, é uma regra da forma

geralmente (X é A)

A ⊂ B temos geralmente(X é B).

A seguir descreveremos algumas regras básicas de inferência.

Regras de conjunção :

X é A

X é B temos X é A ∧ B, onde A ∧ B é a interseção de A e B definidos por $\mu_{A\cap B}(u) = MIN\ [\ \mu_A(u),\ \mu_B(u)]$, $u \in U$.

Produto Cartesiano :

X é A

Y é B temos (X,Y) é A × B, onde (X,Y) representam variáveis que são substituidas por elementos de A e B, e A × B definida por:

$$\mu_{A\times B}(u,v) = MIN\ [\mu_A(u),\ \mu_B(v)],\ u \in U \text{ e } v \in V.$$

Regra de projeção :

(X,Y) é R temos X é $_XR$, sendo a projeção da relação binária R sob o dominio de X, é definida por :

$$\mu(u) = \sup_v \mu_R(u,v),\ u\in U,\ v\in V,$$

onde $\mu_R(u,v)$, é a função de pertinência de R e o supremo é tomado sobre $v \in V$. Para melhor compreensão veja a definição de projeção de relações nebulosas.

Regra de composição :

X é A

(X,Y) é *R* temos Y é A ∘ *R*, a composição da relação unitária A com a relação binária *R*, é definida por

$$\mu_{A \circ B}(v) = \sup_u [\ \text{MIN}\ (\mu_A(u)\ ,\ \mu_R(u,v))].$$

Esta definição é a mesma dada para subconjunto nebuloso condicionado.

2.10 OPERADOR ADITIVO E/OU

A manipulação de dados imprecisos, por sistemas inteligentes, ainda não está totalmente compreendida. Um dos principais pontos de discussão na área da teoria dos conjuntos nebulosos é a definição de operadores de cálculo nebuloso que sejam gerais e representem o comportamento do raciocínio humano. Inicialmente propos-se os operadores min-max, que, para algumas aplicações, mostraram-se úteis. Entretanto não se mostram gerais e consequentemente muitas propostas novas de cálculo nebuloso foram efetuadas.

Podemos definir algumas características essênciais de um operador nebuloso de uso geral, tais como: manipular lógica clássica e lógica nebulosa, permitir um cálculo aditivo (dado dois fatos nebulosos obtermos um dado resultante com maior valor de confiança) e possuir parâmetros que permitam a variação de seu comportamento e consequentemente a modelagem do operador nas mais diversas situações.

Outro problema frequentemente encontrado nas propostas de modelos nebulosos, decorre do fato de inexistir o relacionamento entre o modelo de representação de conhecimento e operadores nebulosos. Ou seja, o modelo deverá ser proposto a partir dos dados adquiridos para a construção do sistema e o operador deve ser configurado baseados nesses dados.

A seguir descreve-se um modelo baseado em grafos nebulosos fornecidos por especialistas e utiliza-se um operador aditivo e/ou único que pode ser configurado de acordo com a situação.

A descrição do conhecimento é efetuada pelo especialista através da construção de grafos nebulosos acíclicos como descrevemos abaixo.

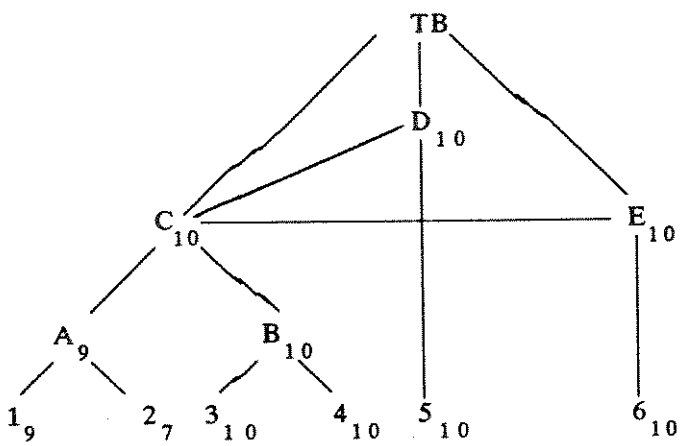


Fig. 2.2 : Grafo de conhecimento sobre Tuberculose

Os nodos inferiores são fatos e estão representados no grafo por números que representam:

1 - Tosse por mais que duas semanas

2 - Febre

3 - Raio X

4 - Presença de bacilo no catarro

5 - Presença de bacilo na cultura

6 - Biópsia sugestiva

Os nodos inferiores são obtidos através da seleção em uma lista de fatos referentes a uma hipótese (neste caso tuberculose). Os fatos são ordenados e colocados no gráfico nebuloso no nível inferior. Estes fatos são conectados de tal forma a descrever os passos efetuados pelo especialista na prova de uma hipótese (grafo procedural) ou conectados de forma a descrever uma dada hipótese (grafos descritivos).

Os nós intermediários (rotulados por letras maiúsculas) dos grafos são manipulações simbólicas de fatos ou de outros nós. O nó superior (rotulado por TB) é o nó final ou hipótese.

Os valores entre [0,10] colocados ao lado dos nós é a adesão que relaciona o nó (fato ou símbolo) à hipótese. Exemplificando, o sintoma febre (nodo rotulado com 2) tem o valor 7 de adesão (importância) ao diagnóstico de Tuberculose. Outro valor manipulado é a confiança ou grau de equivalência entre o fato do grafo (estrutura) e o fato de entrada. Exemplificando, a uma pessoa com 39 grau de febre atribui-se um valor nebuloso 9, que é obtido de um conjunto nebuloso "febre".

Os nós são conectados através de operadores similares ao "e" e o nó final (hipotese) é conectado por operador similar ao "ou".

Baseado em informações obtidas no grafo nebuloso, foi possível definir um único operador aditivo "e/ou", que é uma função de alguns parâmetros contidos no grafo.

O operador é uma função da seguinte forma

$$A(H) : F(a,c,\alpha,n) \not\alpha\rightarrow [0,1]$$

onde "a" é adesão, "c" a confiança, "α" um valor em [0,1] e "n" o número de fatos conectados pelo operador.

A formulação matemática é dada por

$$a(H)=(1-\alpha)* \sqrt[p]{\prod_n a(f_I)*c(f_I)} + \alpha * (1 - \sqrt[p]{\prod_n (1-a(f_I)*c(f_I))})$$

onde $\alpha \equiv [0,1]$ e $a(f_I)$ representa a adesão ao fato I e $c(f_I)$ a confiança associada ao fato I.

Podemos destacar algumas características do comportamento deste operador. Atribuindo valores a "α" temos:

$\alpha = 1$ então "e/ou" → "ao menos um deles" (equivalente ao "ou" clássico);

$0 < \alpha < 1$ então "e/ou" → "muitos deles" (equivalente a lógica nebulosa;

$\alpha = 0$ então "e/ou" → " todos eles" (equivalente ao "e" clássico).

O parâmetro "n" refere-se ao número de elementos que estão sendo conectados. Se o operador for configurado para trabalhar com o valor de "p" menor que "n" (o número de parâmetros) o operador terá um comportamento aditivo (produzirá resultados acima dos valores fornecidos a adesão e confiança). Se o valor dado a "p" é maior que o número de parâmetros o comportamento do operador é restritivo.

A definição do operador aditivo e/ou foi feita baseada nas na forma de representação de conhecimento (grafos) e nas informações manipuladas (adesão e confiança). Através da configuração dos parâmetros do operador aditivo e/ou podemos ter o comportamento: dos operadores da lógica clássica; dos operadores da lógica nebulosa (max-min); de um operador aditivo (utilizado em raciocínio aproximado); de um operador restritivo (onde o valor resultante é menor que o valor dos dados) e de um operador que não necessita de todas os dados para produzir um resultado (raciocínio parcial). Portanto a capacidade deste operador vai muito além dos operadores tradicinais (max-min) da lógica nebulosa e representa um passo na busca de um operador nebuloso geral.

Durante a descrição do modelo do PROLOG-aditivo no capítulo 4 exemplificaremos a utilização deste operador, principalmente no contexto do raciocínio parcial e do aproximado.

CAPÍTULO 3

SISTEMA DE MANIPULAÇÃO DA TEORIA DOS CONJUNTOS NEBULOSOS

Após a teoria dos conjuntos nebulosos ter sido proposta por Zadeh [ZADE65], um grande número de novas aplicações teóricas utilizando este conceito foram efetuadas em áreas tais como teorias dos autômatos, linguagem natural, lógica, tomada de decisão e matemática aplicada. Como consequência desse desenvolvimento teórico tornou-se necessária a criação de ferramentas de programação que permitissem a aplicação e a utilização desta teoria nas mais diversas áreas da engenharia (reconhecimento de padrões, robótica, controle, etc), economia, medicina, etc ..

As primeiras aplicações utilizavam a teoria dos conjuntos nebulosos para solucionar problemas específicos, não havendo intenção da criação de uma ferramenta de programação de uso geral.

As primeiras ferramentas de programação consistiram em sistemas de manipulação de conjuntos nebulosos e foram baseadas em programas construídos para manipular conjuntos ordinários. A ferramenta programação FSDTS [UMAN78] foi uma evolução das anteriores, consistindo de um conjunto de 52 funções orientadas para manipular estruturas nebulosas (conjuntos, relações e L-conjuntos nebulosos) e alguns métodos de inferência. A linguagem FRIL [BALD84] foi inicialmente proposta como um sistema automático de inferência que opera sobre uma base de conhecimento montada como relações matemáticas (relações nebulosas). A capacidade de inferência é obtida através de operações de composição e projeção sobre relações.

O sistema FRIL é sem dúvida alguma a mais bem sucedida ferramenta de programação desenvolvida nesta área. Isto deve-se às frequentes atualizações sofridas pelo sistema, tais como a adoção de uma sintaxe similar à do Prolog (linguagem largamente utilizada em Inteligência Artificial) e a implementação de vários métodos de inferência com dados incertos (lógica nebulosa, programação suportada por lógica, etc). As atualizações, decorrentes de vários anos de desenvolvimentos teóricos e reavaliações, após várias aplicações nas mais diversas áreas, permitiram que a linguagem FRIL alcançasse uma considerável generalização e consequentemente o sucesso comercial.

Dois fatos, até esta fase, mostraram-se fundamentais e necessários na construção das ferramentas:

- poder representar dados nebulosos através dos mais diferentes tipos de estrutura de dados nebulosos;
- possuir mecanismo de inferência nebulosa ou possibilitar a construção de outros modelos (tais como o probabilístico).

Partindo dos fatos acima e com o aparecimento da linguagem PROLOG como linguagem de programação largamente utilizada em Inteligência Artificial, as pesquisas de ferramentas para manipulação de dados incertos tomaram um novo rumo pois tornou-se frequente o desenvolvimento de ferramenta baseada em PROLOG e que manipulam a lógica nebulosa.

Este trabalho seguiu os mesmos caminhos traçados pelas pesquisas desta área. Este capítulo descreve uma ferramenta de programação com as características das linguagem FSDTS [UMAN78] e FRIL [BALD84 e ZHOW84] (conceito inicial) para manipulação da teoria dos conjuntos nebulosos. Nos próximos capítulos será discutido modelos de PROLOG-nebuloso e a implementação de um dos modelos.

3.1 SISTEMA DE MANIPULAÇÃO DE CONJUNTOS E RELAÇÕES NEBULOSAS

Baseado na teoria dos conjuntos nebulosos desenvolveu-se um conjunto de aproximadamente 40 rotinas contendo as principais operações sobre conjuntos e relações nebulosas. As definições teóricas das operações implementadas podem ser encontradas no capítulo 2 e a documentação completa do sistema pode ser encontrada no relatório técnico especialmente montado para este objetivo.

Inicialmente descreve-se a representação formal de conjuntos e relações nebulosas e sua estrutura de dados. Para melhor compreensão dividiremos a descrição das rotinas em módulos. A divisão é feita de acordo com as características das rotinas. A implementação foi efetuada na linguagem de programação C. Portanto algumas características das funções e do sistema estão intimamente ligadas às características desta linguagem. Durante a descrição de como contruir programas, serão montados alguns exemplos que permitirão a compreensão de como utilizar as rotinas do sistema e sua utilidade.

3.1.1 Descrição Formal e Estruturas de Dados

No capítulo anterior definimos conjuntos e relações nebulosas. A seguir formalizaremos conjuntos e relações nebulosa na forma interpretada pelo sistema. Esta forma de representação pode ser considerada padrão, pois é frequentemente encontrada na literatura.

Um conjunto nebuloso será representado por

F= u1/Pert-F(u1), u2/Pert-F(u2), .. , un/Pert-F(un)

onde ui, i=1,...,n representam os elementos no universo do discurso U e Pert-F(ui) representam o valor do grau de pertinência. Portanto assumindo que U={a,b,c,d} um conjunto nebuloso F é representado como

F= a /0.1, b/0.8, d/0.9 .

Uma relação nebulosa será representada por

R = (u1,v1)/pert-r(u1,v1), (u1,v2)/pert-r(u1,u2), ...,
(um,vn)/pert-r(ui,vj)

onde ui, i=1,2,...,m e j=1,...,n representam os elementos de U e V respectivamente, e (ui,vj) o par ordenado de U X V e pert-r(ui,uj) o grau de pertinência. Assumindo U={a,b,c,d}, uma relação R em U x U é representada como

R=(a,b)/0.3, (b,d)/0.8, (d,a)/0.9.

O sistema permitirá somente a utilização de conjuntos discretos. As relações também serão entre conjuntos discretos.

As estruturas do tipo nebulosas são representadas na forma descrita acima. Estas formas são interpretadas e seus elementos e grau de pertinência são inseridos na estrutura de dados.

A definição da estrutura de dados foi efetuada a fim de possibilitar a construção de qualquer estrutura nebulosa. Pode-se implementar L-conjuntos nebulosos, conjunto nebuloso tipo-n e nível-m. Entretanto o sistema está construído para manipular as estruturas de conjuntos e relações nebulosas definidas acima, portanto, em algumas situações, haverá campos de dados desocupados.

A estrutura de dados é definida utilizando-se duas estruturas básicas. A primeira, a qual chamamos de estrutura tipo nebulosa, é descrita abaixo:

```
ESTRUTURA TIPO NEBULOSA
 {
     NOME DO CONJUNTO OU RELAÇÃO
     APONTADOR PARA A PRÓXIMA ESTRUTURA TIPO NEBULOSA
      {
          GRAU DE PERTINÊNCIA OU
          APONTADOR PARA UM CONJUNTO DE PERTINÊNCIA
     }
     APONTADOR PARA O INÍCIO DA LISTA DE ELEMENTOS
 }
```

O campo "nome do conjunto ou relação" conterá o nome da estrutura (nome do conjunto ou relação). Através do campo "apontador para a próxima estrutura tipo nebulosa" cria-se uma lista encadeada contendo as estruturas do mesmo tipo. Atualmente somente duas listas são montadas, uma para conjuntos e outra para relações. O campo "grau de pertinência" permite associar à estrutura um grau de pertinência ou um apontador para um conjunto nebuloso, e finalmente o campo "apontador para o início da lista de elementos" é um apontador para uma estrutura de lista encadeada que representa os elementos.

A segunda estrutura de dados, a qual chamamos de estrutura elemento, representará os elementos e é descrita a seguir.

```
ESTRUTURA ELEMENTO
{
      NOME DO ELEMENTO
      APONTADOR PARA O PRÓXIMO ELEMENTO RELACIONADO NA ESTRUTURA TI
RELAÇÃO
           {
             GRAU DE PERTINÊNCIA
             APONTADOR PARA UM CONJUNTO NEBULOSO
            }
     APONTADOR PARA O PRÓXIMO ELEMENTO DA LISTA
}
```

O campo "nome do elemento" contém o nome do elemento. O campo "próximo elemento relacionado" é um campo utilizado em estruturas tipo relação, para representar relacionamentos entre elementos. O campo "grau de pertinência" contém o grau de pertinência do elemento ou pode conter um apontador para um conjunto nebuloso (usado para representar L-conjunto e conjunto tipo-n), e finalmente o campo "próximo elemento da lista" é um apontador para o próximo elemento da lista encadeada.

O desenho a seguir descreve as estrutura de dados usada para representar relações e conjuntos nebulosos, e o conjunto nebuloso F=a/0.1, b/0.8, d/0.9 e a relação R=(a,b)/0.3, (b,d)/0.8, (d,a)/0.9.

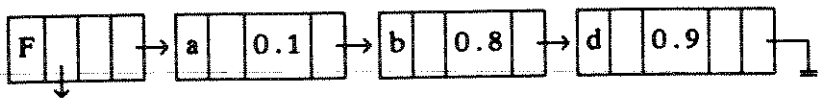
nome do elemento	apontador para o próximo elemento da relação	pertinência ou apontador	apontador para o próximo elemento

a) elementos

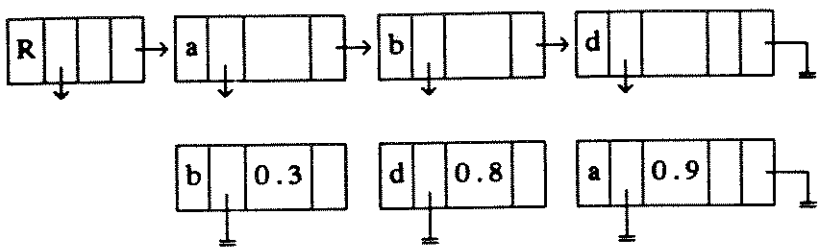
nome da estrutura	apontador para a proxima estrutura nebulosa	pertinência ou apontador	apontador para o início da lista

b) conjuntos

Fig. 3.1: Representação das Estruturas de Dados



a) conjunto nebuloso



b) relação nebulosa

Fig. 3.2 : Representação de conjuntos e relações

Uma das características da estrutura de dados do sistema é a utilização de apontadores. Isto se deve à linguagem C permitir e ter como uma de suas características básicas a manipulação de apontadores.

3.1.2 Características de programação

Este sistema foi construído para ser uma extensão da linguagem de programação C. O programador poderá utilizar todos os comandos e funções disponíveis na linguagem C e possuirá à sua disposição um conjunto de funções e tipos de dados nebulosos.

As funções nebulosas obedecem à sintaxe de definição de funções da lingugem C:

tipo_retorno nome_funcao (lista de parâmetros)

onde a lista de parâmetros e o tipo_retorno são compostos de tipos comumente encontrados na linguagem ou tipos de dado nebulosos que foram acrescentados.

Os tipos nebulosos são definições das estruturas de dados definidas acima. Através do mecanismo de criação de estruturas "struct" e do comando para definição de tipos "typedef" da linguagem C, foi possivel criar tipos de dados nebulosos e utilizá-los na definição de variáveis e funções nebulosas.

Como as estruturas definidas são um tanto complexas, e é improdutivo passar grande quantia de dados como parâmetros de entrada ou saída nas funções, a linguagem C permite a utilização de apontadores, que consiste de uma variável que contém o endereço do início da estrutura e possui meios para dar acesso aos seus campos. Em virtude destas facilidades, um conjunto nebuloso será manipulado como um apontador.

Além de permitir um maior desempenho na comunição de dados entre as funções, a utilização de apontadores facilita a programação (depois de compreendido o processo de manipulação dos apontadores) e permite a passagem transparente de dados entre as funções.

O sistema implementado foi dividido em módulos a fim de permitir ao programador incluir no programa fonte apenas aqueles que contém as funções utilizadas. A linguagem C possui um comando preprocessador com a seguinte sintaxe "#INCLUDE<nome_do_arquivo>" que inclui no programa fonte, em tempo de compilação, o arquivo cujo nome é dado por nome_do_arquivo. Os módulos foram montados de acordo com as características das funções e os módulos são:

Módulo BASE : Neste módulo estão as definições das estruturas dos tipos de dados nebulosos existentes (vide a secão 3.2.1).

Módulo GERAR : Nele estão as rotinas de criação e impressão de conjuntos e relações nebulosas.

Módulo CONJUNTO : Nele estão as rotinas de manipulação de conjuntos nebulosos, operadores de medidas, operadores relacionais.

Módulo RELAÇÃO : Nele estão as rotinas de manipulação de relações nebulosas.

Os módulos BASE e GERAR são os fundamentais, e portanto necessários em qualquer programa que utilize as funções. A seguir estão descritas algumas características das funções e alguns exemplos onde mostra-se algumas das suas aplicações.

A seguir descreve-se um pequeno programa na linguagem C que contém funções para criar (conj), imprimir (impcon) e operar (uniao, complemento e intersecao) sobre conjuntos nebulosos.

```
/*
 *  Este programa computa a fórmula (¬X) v (X ^ Y)
 *  onde X e Y são conjuntos nebulosos e os operadores
 *  ¬,v e ^ são as operações complemento, união e interseção
 */
#include <base.c>
#include <gerar.c>          /*  incluir os arquivos base      */
#include <conjunto.c>       /*  gerar e conjuntos             */
main()
{
/* declaração de variáveis */
nebulosa x, y;
/* dados para os conjuntos */
x = conj("x=a/0.2, b/0.8, d/0.9");
y = conj("y=a/0.7, c/0.5, d/0.5");
impcon(uniao(complemento(x), intersecao(x,y)));
}
```

Inicialmente são colocados os nomes dos arquivos que contém o código fonte das rotinas que devem ser incluídos durante a compilação. Todo programa em C tem ao menos um módulo principal "main()" e as funções que serão usadas devem ser definidas antes. O bloco do módulo "main" é igual ao corpo das funções pois, primeiramente estão as definições das variáveis e em seguida o corpo contendo os comandos a serem executados.

As funções estão documentadas no relatório técnico. Nele encontram-se as caraterísticas, definições, a documentação e o código fonte das funções.

O processo de "compilação" e "linkagem" para geração de código executável é efetuado por um compilador C.

3.1.3 Descrição dos Tipos de Dados e Funções de Entrada e Saída

Para construirmos programas na linguagem C necessitamos definir as variáveis que serão utilizadas. Os tipos de dados são os disponíveis na linguagem C (inteiros, caracteres, real, etc) e foi acrescentado no sistema o tipo de dado nebuloso. Um tipo de dado nebuloso consiste de um apontador para uma estrutura de dados nebulosa como a descrita acima.

Portanto quando definimos

nebulosa alto, baixo;

estamos criando duas variáveis (alto e baixo) do tipo apontador, cujo conteúdo é o endereço onde a estrutura do tipo nebulosa está montada na memória. A definição acima obedece a sintaxe da linguagem C e toda estrutura nebulosa (conjunto ou relação) utilizada por rotinas do sistema deverá ser definida como uma variável do tipo "nebulosa".

Rotinas básicas para a utilização do sistema são as de geração e impressão da estrutura de dados (conjunto e relações nebulosa). As funções de geração são responsáveis pela interpretação de uma sequência de caracteres representando um conjunto ou relação nebulosa na forma descrita na seção 3.2.1. Os elementos e seus respectivos graus são interpretados e inseridos na estrutura de dados. Estas funções retornam o apontador para a estrutura montada. As funções implementadas são: "conj" para gerar conjunto, "rel" para relação e "cp" para gerar relações através de dois conjuntos nebulosos.

As funções de impressão são responsáveis pela recuperação dos dados de uma estrutura (conjunto ou relação), e pela impressão na sintaxe descrita na seção 3.2.1. As funções de impressão são "impcon" para conjuntos nebulosos e "imprel" para relações nebulosas.

A seguir descrevemos como é montado um pequeno programa que utiliza as funções de criação e impressão de estruturas nebulosas.

```
/*
 *      Este programa interpreta e imprime o conjunto "alto" e
 *       relação y
 */
#include <base.c>
#include <gerar.c>
main()
{
  /* declaração de variáveis */
  nebulosa alto, y;
  /* dados para os conjuntos alto e relação y */
  alto = conj("alto=a/0.2, b/0.8, d/0.9");
  y = rel("y = (a,b)/0.7, (b,c)/0.5, (c,d)/0.5");
  /* impressao do conjunto alto e relação y */
  impcon(alto);
  imprel(y);
}
```

No programa são definidas as variáveis nebulosas "alto" e "y". A função "conj" recebe uma cadeia de caracteres, interpreta e coloca os dados na estrutura de dados, e retorna um apontador para a estrutura montada que é atribuído para a variável apontador "alto". Da mesma forma a função "rel" retorna um apontador que é atribuído para a variável "y". As funções de impressão recebem um apontador para a estrutura e imprime no formato da estrutura.

As rotinas para gerar conjuntos podem ser facilmente alteradas para ler conjuntos e relações de um arquivo. Desta forma pode-se, através de um editor de texto, montar uma base de dados com os conjuntos e relações em um arquivo e operar sobre os mesmos. Se for necessário retornar para o arquivo as estruturas criadas ou alteradas, basta direcionar o canal de comunição de saída padrão para o arquivo desejado. A forma correta seria a criação de uma base de dados a ser lida no início do programa e salva no final e que, portanto, pudesse ser utilizada em outras oportunidades.

3.1.4 Rotinas de Manipulação de Conjuntos Nebulosos

Como vimos anteriormente as funções implementadas são divididas em módulos e colocados em arquivos separados. Para facilitar a compreensão dos diferentes tipos das funções existentes, o módulo conjunto foi redividido em quatro grupos de acordo com as características das funções.

a) Rotinas de Manipulação de múltiplos conjuntos nebulosos

Neste grupo estão as rotinas comumente usadas para manipular conjuntos nebulosos. Estas funções distinguem-se por receberem dois apontadores para estrutura que contém conjuntos nebulosos como parâmetros de entrada, retornando um apontador para uma estrutura conjunto nebuloso resultante da operação.

Abaixo estão os nomes das funções deste grupo; seu nome no sistema é colocado entre parênteses.

União de conjuntos nebulosos (uniao).
Interseção de conjuntos nebulosos (intersecao).
Soma disjuntiva de conjuntos nebulosos (somadis).
Diferença básica de conjuntos nebulosos (bdif).
Soma básica de conjuntos nebulosos (bsoma).
Produto de conjuntos nebulosos (produto).
Soma algébrica de conjuntos (somalgeb).
Diferença de conjuntos nebulosos (diferenca).

b) Rotinas de Manipulação de um único conjunto nebuloso

Este grupo se diferencia do anterior por ter apenas um conjunto nebuloso como parâmetro de entrada, por operar sobre o mesmo e retornar um conjunto nebuloso ou ordinário.

Abaixo estão os nomes das funções deste grupo; seu nome no sistema é colocado entre parênteses.

Complemento de um conjunto nebuloso (complemento).

Conjunto Vulgar mais próximo de um conjunto nebuloso (rvulgar).

Conjunto vulgar alfa mais próximo (rvulgalfa).

Cutt (Cutt).

Dilatação de conjunto nebulosos (dilatacao).

Concentração de conjunto nebuloso (concentracao).

Exponenciação de conjunto nebuloso (exponenciacao).

Normalização de conjunto nebuloso (norma).

Sop (sop).

c) Rotinas dos Operadores de medidas

São operações para cálculo da distância entre conjuntos nebulosos e medidas dos conjuntos nebulosos. O parâmetro retornado é um valor real que denota o valor resultante da medida ou do cálculo da distância.

Abaixo estão os nomes das funções deste grupo; seu nome no sistema é colocado entre parênteses.

Cardinalidade de um conjunto nebuloso (cardinalidade).

Distância Hamming entre conjuntos nebulosos (hamming).

Distância Hamming relativa entre conjuntos nebulosos (dishammingrel).

Distância Euclidiana entre conjuntos nebulosos (euclides).

Distância Euclidiana relativa entre conjuntos nebulosos (euclidesrel).

Índice linear de nebulosidade (linear_fuzzi).

Índice quadrático de nebulosidade (ind_quadr_neb).

d) Rotinas de Operadores relacionais

Este grupo de operadores destaca-se por efetuar a comparação de conjuntos nebulosos. Esta função compara os elementos do primeiro conjunto nebuloso (primeiro parâmetro de entrada) com os elementos do segundo conjunto nebuloso (segundo parâmetro de entrada). As rotinas retornam 0 se a comparação for falsa e 1 se a comparação é verdadeira.

Abaixo estão os nomes das funções deste grupo; seu nome no sistema é colocado entre parênteses.

Subconjunto nebuloso (subconjunto).

Igual (igual).

Igualdade entre conjuntos (igualdade).

Disjunção entre conjuntos nebulosos (disjunto).

3.1.5 Rotinas de manipulação de Relações nebulosas

As funções que manipulam relações estão contidas no módulo relação. As rotinas são divididas em dois grupos, de acordo com os parâmetros de entrada, para permitir melhor compreensão. A seguir descrevem-se os dois grupos:

a) Rotinas de manipulação de relação nebulosa

Esta categoria de operadores distingue-se por receber como parâmetro de entrada uma relação nebulosa (as funções C-image e condicionado recebem uma relação e um conjunto nebuloso) operar sobre a mesma e retornar um conjunto nebuloso ou uma relação nebulosa.

Abaixo estão os nomes das funções deste grupo; seu nome no sistema é colocado entre parênteses.

Projeção horizontal de uma relação nebulosa (rprojh).

Projeção vertical de uma relação nebulosa (rprojv).

Subconjunto nebuloso condicionado (condicionado).

C-Image (cimage).

Inversão de relação nebulosa (inverte).

b) Manipulação de relações nebulosas

Este grupo de operadores distingue-se por receber como parâmetros de entrada duas relações nebulosas operar sobre as mesmas e retornar uma relação nebulosa. Estas funções são de composições de relações nebulosas.

Abaixo estão os nomes das funções deste grupo; seu nome no sistema é colocado entre parênteses.

Composição max-min de relações nebulosas (composicao).

Composição min-max de relações nebulosas (composmimax).

Composição max-produto de relações nebulosas (composmaxprod).

3.2 EXEMPLOS DE APLICAÇÕES

A seguir serão exemplificadas algumas aplicações das funções descritas acima e serão destacadas as áreas da teoria dos conjuntos nebulosos em que as funções podem ser utilizadas.

O conceito de variáveis linguísticas definido no capítulo anterior é uma das áreas em que podemos aplicar as funções de operações sobre conjuntos. Podemos representar conjuntos primários ("alto", "baixo", etc..), utilizar e criar modificadores ("muito", "aproximadamente", etc..) e conectores ("e" e "ou" equivalentes às funções interseção e união respectivamente).

Dado um conjunto nebuloso

alto = 160 / 0.5, 170 / 0.7, 175/ 0.8, 180/0.9

podemos obter os conjuntos nebulosos referentes às variáveis linguísticas "muito alto" e "aproximadamente alto" utilizando as funções dilatação e concentração (equivalente aos modificadores "muito" e "aproximadamente"). O programa é construído da seguinte forma.

```
#include <base.c>
#include <gerar.c>
#include <conjunto.c>

 /*
  *  Implementação de variáveis linguísticas
  */
main()
{
 /* declaração de variáveis */
 nebulosa alto;
 /* dados para os conjuntos x e y */
 alto = conj("alto=160/0.5, 170/0.7, 175/0.8, 180/0.9");
 /* criação do conjunto "muito alto" */
 impconj(concentracao(alto));
 /* criação do conjunto "aproximadamente alto" */
 impconj(dilatacao(alto));
 }
```

A teoria da possibilidade é largamente utilizada em aplicações que manipulam dados nebulosos.

As funções de manipulação de conjuntos podem ser utilizadas em algumas implementações da teoria da possibilidade (a definição pode ser encontrada no capítulo 2).

O cálculo da possibilidade é dado por

$$\text{Poss } (X \text{ e } F) = P(F)$$
$$= \sup [\ M(u) \ \&\ p(u)]$$

O exemplo utilizado é o mesmo do capítulo anterior. Dada uma distribuição Ausencia_Bob = {15/0.3, 16/0.5, 17/1.0, 18/0.5 } e um conjunto nebuloso "meio_do_mes" = { 1.0/15, 0.6/16, 0.7/17,0.4/18 } podemos obter a possibilidade da ausência de Bob no meio do mês e a implementação é dada abaixo.

```
#include <base.c>
#include <gerar.c>
#include <conjunto.c>
 /*
  * Implementação da Teoria da possibilidade
 */
main()
{
  /* declaração de variáveis */
  nebulosa ausencia_bob, meio_mes;
  float poss_aus_bob_meio_mes;
  /* dados para os conjuntos ausência_bob e meio_mes  */
  ausencia_bob = conj("ausencia_bob=15/0.3, 16/0.5, 17/1.0, 18/0.5");
  meio_mes = conj("meio_mes=15/1.0, 16/0.6, 17/0.7, 18/0.4");
  pos_aus_bob_meio_mes = max(intesecao(ausencia_bob,meio_mes));
   printf(" Possibilidade da ausencia de Bob no meio do mes = %f",
          pos_aus_bob_meio_mes);
}
```

Se desejarmos obter, através do conceito medidas de necessidade, a ausência de Bob no meio do mês, basta inserir no programa o seguinte comando :

```
pos_aus_bob_meio_mes = min(uniao(complemento(ausencia_bob) ,meio_mes));
```

Ainda dentro da teoria da possibilidade foram definidos alguns métodos de inferência que estão descritos na seção 2.10.1 no capítulo 2. O sistema permite que a implementação desses métodos possa ser efetuada. As regras de inferência conjunção e produto cartesiano equivalem às funções interseção (intersecao) e produto cartesiano (cp).

A regra de inferência projeção equivalem às funções projeção vertical e horizontal. O módulo em que se encontram essas funções é o relação. A seguir se constrói um programa para exemplificar a utilização dessas funções. Dada uma relação nebulosa que relaciona os conceitos peso (dado em kilogramas) e altura (dada em centímetros):

	50	60	70	80
160	0.4	0.5	0.7	0.8
170	0.3	0.4	0.5	0.7
180	0.2	0.3	0.4	0.5

Podemos obter um conjunto, o qual denominamos p_alto, cujos elementos estão contidos no conjunto alto e outro conjunto, o qual denominamos p_peso, cujos elementos estão contidos no conjunto peso, através da projeção horizontal e vertical da relação. Os conjuntos obtidos pela aplicação das funções são

p_peso = 50/0.4, 60/0.5, 70/0.7, 80/0.8 e
p_alto = 160/0.8, 170/0.7, 180/0.5

```
/*
 *  Implementação da projeção vertical e horizontal
 *  em relações
 */
#include <base.c>
#include <gerar.c>
#include <relacao.c>
main()
{
 /* declaração de variáveis */
 nebulosa peso_altura;
 /* dados para a relação peso_altura */
  peso_altura    = rel("peso_altura =    (50,160)/0.4,    (60,160)/0.5,  (70,160)/0.7,
                                (80,160)/0.8,          (50,170)/0.3,          (60,170)/0.4,
                                (70,170)/0.5,          (80,170)/0.7,          (50,180)/0.2,
                                (60,180)/0.3, (70,180)/0.4, (80,180)/0.5");
  /* impressão da relação */
 imprel(peso_altura);
  /* cálculo da projeção vertical e horizontal */
 printf(" projecao horizontal ");
 impcon(projh(peso_altura));
 printf(" projecao vertical ");
 impcon(projv(peso_altura));
}
```

A composição é uma das mais importantes regras de inferência da teoria dos conjuntos nebulosos. Dado um conjunto nebuloso e uma relação nebulosa podemos obter um novo conjunto nebuloso resultante da composição do conjunto e da relação. As funções que implementaram estes conceitos são condicionado e Cimage. Utilizando a relação peso_altura descrita no exemplo anterior e dado um conjunto nebuloso que atribui a cada elemento (dados em centímetros) um grau de pertinência:

altura = 160/0.5, 170/0.6, 180/0.7.

Podemos obter pela aplicação da operação de composição condicionada um conjunto induzido que representa o "peso" que neste caso é dado por:

c_peso = 50/0.4, 60/0.5, 70/0.5, 80/0.6.

Montando o programa temos

```
/*
 * Utilização da função Condicionado
 */
#include <base.c>
#include <gerar.c>
#include <relacao.c>
main()
{
 /* declaração de variáveis */
 nebulosa peso_altura, altura;
 /* dados para para a relação peso_altura */
  peso_altura    = rel(" peso_altura =    (50,160)/0.4,    (60,160)/0.5,
                  (70,160)/0.7,    (80,160)/0.8,    (50,170)/0.3,    (60,170)/0.4,
                  (70,170)/0.5,    (80,170)/0.7,    (50,180)/0.2,    (60,180)/0.3,
                  (70,180)/0.4,  (80,180)/0.5" );
 altura  = conj("altura=160/0.5,    170/0.6,    180/0.7").
 printf(" conjunto condicionado resultante = ");
 impconj(condicionado(peso_altura));
}
```

Podemos efetuar muitas outras aplicações destas funções. Entretanto deve-se deixar claro que as funções foram montadas para conjuntos discretos, o que leva a restringir o escopo de suas aplicações.

Vimos que os elementos básicos de operações são os conjuntos e relações. Perde-se eficiência quando desejamos obter o resultado da aplicação de uma operação sobre um dado elemento de um conjunto, pois as operacões são efetuadas em todos os elementos levando a um gasto de tempo de processamento desnecessário. Portanto, este modelo deve ser aplicado a problemas em que os elementos básicos sejam conjuntos e relações discretas e certamente não se desejando velocidade na obtenção dos resultados.

CAPÍTULO 4

PROLOG NEBULOSO

A criação do PROLOG foi o primeiro passo visando a utilização do predicado de primeira ordem como paradigma de uma linguagem de programação. A linguagem foi projetada e implementada por Colmenrauer e seu grupo de Inteligência Artificial (GIA), na Universidade de Marselha, onde foi escrito o primeiro interpretador na linguagem Algol-W [CASA et alli] . Novas implementações seguiram-se, visando melhorar o desempenho. A linguagem passou a atrair um amplo interesse quando foi implementada, na Universidade de Edinburg, uma versão eficiente que ficou conhecida por DEC-10 Prolog ou Edinburg Prolog.

Nos últimos anos a linguagem PROLOG difundiu-se, principalmente após ter sido anunciada a sua utilização no projeto japônes de um computador de quinta geração. Atualmente a linguagem tem sido utilizada em aplicações nas áreas de:

- Engenharia de Software,
- Bancos de Dados,
- Manipulação simbólica de fórmulas matemáticas,
- Prova automática de teoremas,
- Construções de Compiladores.

Entretanto é na Inteligência Artificial (IA) que o PROLOG tem sido largamente utilizado, nas sub-áreas de engenharia do conhecimento, processamento de linguagem natural, sistemas especialístas, etc ..

Nos trabalhos da IA frequentemente deparamos com aplicações que necessitam manipular dados nebulosos (informações incompletas, imprecisas e incertas). Comumente nos trabalhos que utilizam o PROLOG, criam-se cláusulas para manipular essas informações nebulosas. Pode-se notar nas implementações, que as cláusulas ou predicados construídos para esse propósito não são manipulados naturalmente pelos mecanismos do PROLOG, além de serem difíceis de se construir. Em adição sua realização é deselegante e de difícil compreensão pelo programador.

A lógica nebulosa tem mostrado ser um conceito teórico útil na manipulação e inferência com dados nebulosos. Para tornar o PROLOG capaz de operar informações nebulosas, foram efetuados estudos para estender conceitos da lógica binária que fundamentam o PROLOG, para a lógica nebulosa.

Um dos fatos a destacar é que os conceitos da lógica nebulosa contêm grande parte das definições da lógica binária, (esta comparação foi efetuada no capítulo 2) e consequentemente abrange os predicados lógicos de primeira ordem. Portanto o PROLOG pode ser visto como um caso especial de um PROLOG-nebuloso onde todos os dados são absolutamente verdadeiros (ou falsos).

O trabalho de R. C. T. Lee [LEE72] discutiu a relação entre a lógica nebulosa e a lógica binária no contexto da lógica de primeira ordem e do princípio da resolução. Este foi o primeiro passo tanto para a definição da resolução nebulosa quanto para provar que a inferência em lógica nebulosa pelo princípio da resolução é significativo quando o valor verdade de todas as variáveis são tomadas no intervalo (0.5, 1].

A partir destas definições teóricas foram efetuados várias implementações de linguagens de programação que manipulam conceitos nebulosos com sintaxe PROLOG. Os principais exemplos são: Fuzzy Prolog [MUKA87 et alli] e Fprolog [MART87 et alli].

O Fprolog é um interpretador que possui as mesmas características do PROLOG convencional. A sintaxe é semelhante à do Micro PROLOG e sua diferença consiste em que os fatos nebulosos possuem o seu valor de grau de pertinência representado. Os fatos com grau de pertinência igual a 1 não necessitam ter este representado permitindo desta forma a reutilização de todos os programas construídos para o PROLOG. Os operadores utilizados neste modelo são os de mínimo e máximo. Portanto ao utilizarmos este interpretador os resultados serão dados por valores no intevalo (0,5,1]. O PROLOG nebuloso(PRONEB) implementado, que será descrito no próximo capítulo, posssui as mesmas características do Fprolog, porém com a sintaxe do PROLOG-10 ou Edinburg.

Outro modelo descrito na literatura é o Fuzzy Prolog, onde se encontram algumas características que o diferenciam do PROLOG convencional e alguns conceitos recentes da lógica nebulosa. Nas regras foi inserido o conceito de peso (valor que liga premissa e conclusão). O cálculo nebuloso utiliza lógica negativa e positiva para manipulacão do dado, no entanto os resultados são valores em [0,1]. Outro fator importante a ser destacado neste trabalho é que um fato nebuloso sempre unifica, pois o grau de pertinência pode ser obtido por uma função ou pela interpolação dos dados contidos na base de dados. Outras características do Fprolog e Fuzzy Prolog serão discutidas durante a descrição destes modelos de PROLOG-nebuloso.

Atualmente, com a evolução teórica da lógica nebulosa, novos cálculos têm sido propostos, como, por exemplo, o cálculo aditivo e/ou (veja capítulo 2). Discutiremos neste capítulo, a construção de ferramentas de programação baseadas nos mecanismos do PROLOG e que utiliza o operador aditivo e/ou.

A tendência para a criação de ferramentas de programação nebulosa baseadas no paradigma lógico é consequência de boa parte dos pesquisadores em I.A. terem adotado o PROLOG como linguagem. Dado que as limitações do PROLOG são grandes e que os trabalhos recentes de I.A. têm sido direcionado para a utilização do paradigma orientado por objetos, certamente estes fatos irão influênciar os futuros projetos de ferramentas de programação nebulosa. Portanto as pesquisas para a criação de ferramentas que manipulam lógica nebulosa tem sido direcionadas para linguagens que possuam características da linguagem PROLOG e, futuramente, para conceitos de orientação por objetos.

Neste capítulo iremos definir as caraterísticas teóricas dos modelos de PROLOG-nebuloso.

4.1 DESCRIÇÃO DO PROLOG

Nesta seção descrevem-se os conceitos de programação em lógica e as características de um PROLOG convencional, o que facilitará a conceituação e descrição dos modelos de PROLOG-nebuloso. Descreve-se como efetuar a criação e as inconveniências da construção de predicados PROLOG que manipulam a lógica nebulosa.

4.1.1 Programação em lógica e PROLOG

A linguagem PROLOG deve ser entendida como a implementação da idéia de programação em lógica para um subconjunto de cláusulas definidas.

A elaboração de programas em PROLOG, consiste na construção de uma base de dados contendo informações relevantes a respeito do assunto a ser questionado. A base deve ser montada por um conjunto de cláusulas na forma de Horn. Uma cláusula de Horn pode ser uma cláusula definida ou cláusula objetivo.

Uma cláusula definida, sobre um alfabeto A de primeira ordem, é uma expressão da forma

"L :- M_1...M_m" ou da forma "L :- ",

onde L, M_1,.., M_m são literais positivos sobre A, onde o literal L é a cabeça e M_1,..,M_m formam o corpo da cláusula. Esta representação pode ser vista como "L$\vee \neg M_1 \vee .. \vee \neg M_m$".

Uma cláusula objetivo, sobre um alfabeto A de primeira ordem, é uma cláusula vazia ou é uma expressão da forma " $M_1..M_m$", onde $M_1..M_m$ são literais positivos sobre A. Estes literais formam o corpo da cláusula e sua representação também pode ser vista como "$\neg M_1 \vee .. \vee \neg M_m$".

a) Sistema de Resolução PROLOG

Para inferirmos informações da base de dados criou-se um sistema formal de resolução. O sistema formal de resolução trabalha exclusivamente com cláusulas e contém apenas uma regra de inferência, chamada de regra da resolução, que gera uma nova cláusula a partir de duas outras. Dado um conjunto S de cláusulas e uma cláusula C, uma dedução de C a partir de S neste sistema formal consiste na obtenção de uma sequência de cláusulas terminando em C e gerada aplicando-se repetidamente a regra de resolução. Uma refutação a partir de S é a dedução da cláusula vazia (□) a partir de S. A regra da resolução é definida de tal forma que S é inválida se e somente se existir uma refutação a partir de S.

A regra da resolução é a combinação de uma adaptação do Modus Ponens (veja cap. 2) e do processo de unificação. No Modus Ponens define-se que dado duas cláusulas na forma de A e ¬A ∨ B podemos derivar B. Exemplificando a utilização do Modus Ponens, seja um conjunto de cláusulas (equivalente ao conjunto S citado acima) numeradas de 1 a 5, podemos aplicando repetidas vezes o Modus Ponens obter a cláusula vazia (□) que consiste em uma refutação a partir do conjunto de cláusulas

Aplicando o Modus Ponens para o conjunto de cláusulas dado abaixo (numeradas de 1 a 5):

1. A ∨ ¬B ∨ ¬C
2. B ∨ ¬D
3. ¬A
4. C
5. D.

Podemos derivar as cláusulas:

6. A ∨ ¬C ∨ ¬D a partir da regra 1 e 2
7. A ∨ ¬D a partir da regra 6 e 4
8. A a partir da regra 7 e 5
9. □ a partir da regra 8 e 3

O outro passo da regra de resolução é o processo de unificação consiste em tornar idêntico os literais de um conjunto de cláusulas através da substituição de variáveis por termos. Este processo deve ser feito de forma que a substituição seja a mais simples possível e encontremos um unificador mais geral e que não bloqueie as futuras possíveis substituições.

A unificação pode ser aplicada para tornar idênticos um par de literais complementares oriundos de cláusulas diferentes ou tornar idênticos literais de uma mesma cláusula.

A combinação dos dois fatores, Modus Ponens e unificação, que compõe a regra da resolução é exemplificada a seguir.

Seja P o seguinte conjunto de cláusulas:

1. chama(a,b)
2. usa(b,e)
3. ¬chama(X,Y) v depende(X,Y)
4. ¬usa(X,Y) v depende(X,Y)
5. ¬depende(X,Z) v ¬depende(Z,Y) v depende(X,Y)
6. ¬depende(a,e)

Aplicando a regra de resolução obtemos

7. depende(a,b) a partir de 1, 3 com <X/a, Y/b>
8. depende(b,e) a partir de 2, 4 com <X/b, Y/e>
9. ¬depende(b,Y) v depende(a,Y) a partir de 5, 7 com <X/a, Z/b>
10. depende(a,e) a partir de 8, 9 com <Y/e>
11. □ a partir de 6 e 10

A regra 7 é gerada ao aplicarmos o Modus Ponens as regras 1 e 3 e substituirmos (passo da unificação) a variável X pelo termo "a" e a Y por "b".

Nos exemplos acima foi utilizado o procedimento de refutação baseado em resolução e que consiste em dado um conjunto qualquer de cláusulas, procura sistematicamente derivar a cláusula vazia utilizando apenas a regra da resolução e tendo como ponto de partida uma cláusula do conjunto. Uma análise de procedimento de refutação baseado em resolução pode ser encontrado em [CASA87 et alli]

As propriedades da resolução, mencionadas acima, asseguram que a ordem em que os literais são cancelados não influênciam o resultado final da prova. Portanto o número de passos até a prova do objetivo é dependente dos literais escolhidos.

Podemos representar os possíveis caminhos percorridos para obter todas as soluções de uma base de dados, através de uma árvore de busca. Na árvore os nodos intermediários são cláusulas resultante da substituição do sub-objetivo pelo corpo da cláusula, de mesmo nome e parâmetros, contida na base de dados. Os nodos finais ou folhas iguais a vazio (□), indicam que não há novas cláusulas a serem executadas e o objetivo foi provado, caso contrário o objetivo é insolúvel. A seguir temos em b) um conjunto de cláusula e em a) a representação da árvore de busca correspondente.

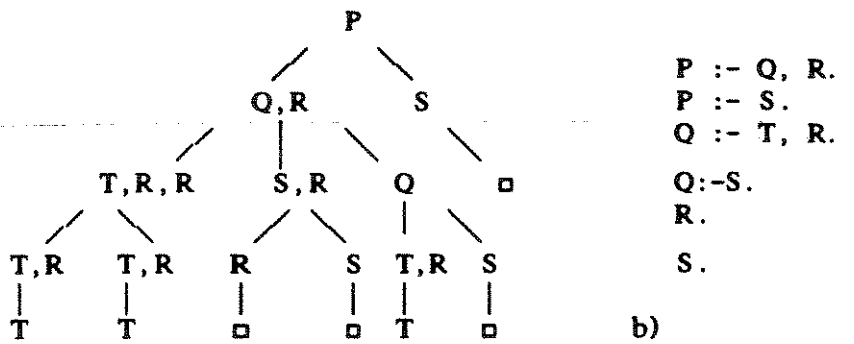


FIG 4.1:Árvore de busca a) para a base de dados
b) descrita ao lado

Os detalhes de implentação de uma base de dados pode ser vista na descrição da implementação do PROLOG-nebuloso. Pode-se antecipar que os dados manipulados pelo PROLOG são construídos na forma de listas. Por exemplo, uma cláusula objetivo ":-A, B, C." é uma lista e é representada como na figura abaixo.

FIG 4.2 : Representação de uma cláusula na forma de lista.

b) Estratégia de Busca

A estratégia usada na implementação do PROLOG consiste em percorrer a árvore de busca em profundidade e tentar provar o primeiro sub-objetivo da lista de consulta, se o sub-objetivo falhar criou-se um mecanismo para retornar ao último nodo de substituição e reiniciar o processo de busca. Portanto, através deste mecanismo, obtemos a primeira solução encontrada na base de dados. A seguir mostraremos detalhadamente este processo.

Podemos considerar uma consulta como a chamada de um conjunto de procedimentos, como vemos abaixo. Quando efetuamos uma consulta

$$:- A_1,...,A_2,A_n. ,$$

inicialmente invocamos o primeiro procedimento sub-objetivo da lista (A_1), e em seguida pesquisa-se na base de dados, se há um procedimento A_1, com o mesmo nome e número de parâmetros, para unificar.

Se o processo de unificação do procedimento sub-objetivo (por exemplo A_1) tiver sucesso, então existe ao menos uma cabeça de cláusula contida na base igual a cláusulas do sub-objetivo A_1 e é assegurado pelo processo de unificação que foram efetuadas as substituições apropriadas. Um novo objetivo é obtido substituindo o literal A_1 contido na consulta pelo corpo da cláusula unificada contida na base de dados(se o corpo não é vazio). Então seja

$$A_1 :- B_1, B_2, ..., B_m.$$

a cláusula contida na base de dados, a nova lista objetivo fica da seguinte forma

$$:- < B_1, B_2, ..., B_m, A_2, ..., A_n > \theta$$

onde θ é o unificador mais geral, ou seja especifica as substituições mais simples possíveis das variáveis do sub-objetivo A_1 e da cabeça da cláusula A_1 contida na base.

A consulta terá sucesso quando todos os procedimentos contidos na lista objetivo forem executados ou a lista objetivo reduzir-se a lista vazia.

Quando um sub-objetivo, por exemplo B_1 não unificar com nenhuma cláusula cabeça contida na base de dados, o sub-objetivo falha e o sistema retorna ("backtrack") ao ponto da última unificação (neste caso A_1). Isto é acompanhado por descartar as substituíções θ criadas pela última unificação A_1, então A_1 é novamente o sub-objetivo a ser buscado na base (a pesquisa é efetuada a partir da última cláusula unificada por A_1). Caso o "backtrack" ocorra e não exista um novo procedimento na lista de objetivo, a consulta falha; isto é, a consulta não é provada pelo sistema PROLOG.

Para ilustrar, mostraremos os passos executados quando efetuamos a consulta de "P" utilizando a base de dados descrita na figura 4.1. A consulta, assim como todas as estruturas PROLOG é construída na forma de lista, então a lista de consulta é dada por ":-P.". Toma-se o primeiro sub-objetivo da lista de consulta, neste caso é "P", e em seguida pesquisa-se na base de dados a primeira cabeça de cláusula para unificar (neste exemplo P unificará com P :- Q, R). Como verificamos anteriormente o processo de unificação assegura que serão efetuadas as substituições apropriadas e que o sub-objetivo P e a cabeça da cláusula são idênticas. A lista da consulta é alterada para ":- Q, R" através dasubstituição do literal pelo corpo da cláusula unificada na base de dados. Portanto, o próximo objetivo a ser executado é "Q". Novamente temos duas cláusulas Q na base, porém o mecanismo busca a primeira solução (Q :- T, R) e a lista de objetivos é alterada novamente para (":- T, R, R"). O sub-objetivo a ser executado é o "T" que não pode ser unificado com cláusulas da base de dados e a unificação falha. Nesta situação o sistema retornará (também conhecido como "backtrack") ao ponto em que foram efetuadas as mais recentes unificações, que neste caso é a substituíção de "Q" por Q :- T, R. Como existe uma outra possível unificação

para Q (Q :- S) a lista da consulta será novamente alterada para " :- S, R". Serão tomados na sequência os sub-objetivos S e R (que são fatos e nós terminais) que serão unificados com as cláusula da base de dados e obteremos a lista de objetivos vazia que significa que a consulta obteve sucesso. A árvore de busca do PROLOG deste exemplo é descrita abaixo:

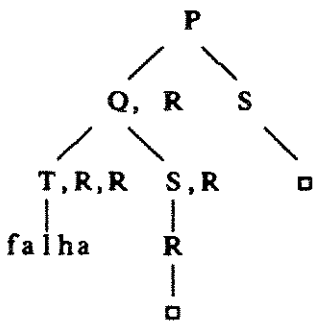


FIG 4.3: Árvore de Busca PROLOG para a consulta "P" à base de dados dada na figura 4.1.

O mecanismo de estratégia de busca dá-nos a primeira solução possivel, entretanto no PROLOG temos outros mecanismos para buscar a próxima solução e desta forma obter todas as soluções existentes na árvore de busca.

c) Mecanismos Extra Lógicos

Um mecanismo extra lógico, encontrado nas diversas implementações do PROLOG, é o "cut". Este mecanismo permite a poda de ramos da árvore de busca das soluções. Isto significa a eliminação de soluções. Esta ferramenta é colocada a disposição do programador, cabendo a ele escolher os pontos de poda. Se usada indevidamente pode eliminar soluções importantes.

Se a cláusula da consulta possuir variáveis, e a consulta tiver sucesso, as variáveis conterão o resultado final das unificações e substituições ocorrida durante a prova da cláusula.

4.1.2 A Linguagem PROLOG

Na seção anterior descreve-se algumas das principais características dos mecanismos de funcionamento da linguagem PROLOG. Nesta são abordados aspectos da sintaxe e semântica do PROLOG. A sintaxe que sera descrita é a do PROLOG-10 ou Edinburg PROLOG e antecipa a descrição sintática do protótipo de PROLOG-nebuloso

implementado.

a) Semântica

A linguagem PROLOG possui um paradigma declarativo que é inerente à lógica, e que consiste em expressarmos declarativamente a estrutura lógica dos problemas através de fatos e regras. Entretanto, podemos ter dentro da linguagem PROLOG outras duas representações semânticas.

A interpretação procedimental no PROLOG permite ao programador identificar e descrever um problema em subproblemas reduzindo a complexidade do mesmo, e a implementação dos subproblemas é feita através da definição de uma série de chamadas de procedimentos.

Finalizando, a semântica operacional que reintroduz a idéia de controle de execução que é irrelevante do ponto de vista da semântica declarativa, através do controle de execução tanto da ordem das cláusulas em um programa PROLOG, quanto das fórmulas atômicas em uma cláusula PROLOG. Através da semântica operacional podemos simular na linguagem PROLOG os comandos de controle das linguagens convencionais (ex."for" , "repeat" , etc).

Um programa PROLOG representa instruções e dados através de um único formalismo que são as cláusulas. Um programa PROLOG é composto basicamente da :

- declaração de fatos sobre objetos e seus relacionamentos,
- definição de regras sobre objetos e seus relacionamentos,
- questões sobre objetos e seus relacionamentos.

Programar em PROLOG consiste em montar os fatos e regras dentro do escopo de um problema. Um sistema PROLOG possui mecanismos para interpretar as regras e fatos, montar a base de dados e possibilitar a execução da inferência sobre esses fatos.

Os fatos são usados para relacionar objetos. Abaixo temos um fato representando a frase "João gosta de Beatriz"

```
gosta(joao,beatriz).
```

onde "gosta" é o relacionamento e "joao" e "beatriz" são os objetos. Os nomes dos objetos e relacionamentos devem ser iniciados por letras minúsculas. Caso o objeto seja variável a primeira letra deve ser maiúscula. A palavra que define o relacionamento é geralmente colocada primeira ("gosta"), e os objetos relacionados a seguir e entre parênteses ("(joao,beatriz)") e finalizando por um ponto.

Regras são usadas quando se deseja representar um fato que dependa de um grupo de outros fatos. Uma regra é usada para expressar definicões do tipo:

X é pássaro se : X é um animal, e Xtem pena.

Na sintaxe do Prolog-10, a regra acima fica:

passaro(X) :- animal(X), tem_pena(X).

Uma regra Prolog consiste em cabeça e corpo conectados pelo simbolo ":-" (que pode ser traduzido como "se") e é finalizada por um ponto. A cabeça e o corpo são compostos por fatos. O escopo das variáveis em um programa PROLOG restringe-se à regra que a utiliza. No exemplo acima a variável X tem seu escopo restríto à regra. A cabeça da regra consiste do fato que a regra visa definir. Quando definimos uma regra, necessitamos definir os fatos ou regras que compõe o corpo da regra. O corpo na regra acima " animal(X), tem_pena(X)" descreve a conjunção dos objetivos que devem ser satisfeitos para que a cabeça seja verdade. Internamente podemos ver o corpo da regra da seguinte forma: ','(animal(X), tem_pena(X)), onde ',' simboliza o operador conjunção que possui dois operandos. Para que a cabeça seja verdade a conjunção exige que seus dois operados "animal(X)" e "tem_pena(X)" sejam verdadeiros.

b) Sintaxe

A sintaxe em que foram descritas as regras e fatos nos exemplos da seção anterior é a mesma do Prolog-10. A seguir descreve-se a sintaxe do Prolog-10.

Os termos são as unidades básicas do PROLOG. Um termo pode ser uma constante, variável ou estrutura.

Uma constante é a forma usada para denotar um objeto especifico ou seus relacionamentos. Existem dois tipos de constante: átomos ou inteiros. Exemplos de átomos são:

gosta joao beatriz passaro animal tem_pena

e exemplos de inteiros são:

0 1 999 8800 8989.

A linguagem PROLOG pura não manipula números. As implementações atuais possuem mecanismos para manipulações numéricas, inclusive números reais.

Os termos que representam variáveis são caracterizados por iniciar com a letra maiúscula ou pelo caracter "_". Um exemplo da utilização de variáveis é dada abaixo :

```
irma_da(X,Y) :- femea(X), pai(X, M, F),
                pai(Y, M, F).
```

Usa-se as variáveis X, Y, M, F para auxiliar a criação de uma regra que represente a afirmação "a pessoa (X) é irmã da pessoa (Y)".

O terceiro tipo de termo, que pode ser utilizado na construção de programas são as estruturas. Estrutura é um objeto simples que contém uma coleção de outros objetos, chamados componentes. Um exemplo de estrutura é o índice de uma biblioteca.

propriedade(eduardo, livro(o_alquimista,autor(paulo_coelho))).

Esta estrutura contém vários componentes ou objetos: o nome do autor, o título do livro e o proprietário. Uma estrutura permite que as informações sejam tratadas como um objeto simples e não como entidades separadas. Uma estrutura em PROLOG é especificada por seu "functor", que no exemplo acima é "propriedade", e seus componentes são colocados entre parênteses e separados por vírgulas.

A sintaxe de um fato e de uma estrutura são equivalentes. O predicado usado em fato e regra é o "functor" na estrutura. Os argumentos de um fato ou de uma regra são os componentes na estrutura.

A sintaxe da linguagem PROLOG básica pode ser opcionalmente estendida através da declaração de operadores PROLOG representados por átomos. Esta extensão melhora a legibilidade dos programas, sem acrescentar poder adicional em termos de expressão da linguagem. Uma cláusula unitária

chama(a,b).

com a definição de "chama" como um operador infixo, pode ser codificada como

a chama b.

Um operador é declarado, antes da sua utilização, numa cláusula da forma :

op(nome, tipo, prior).

onde:

nome : átomo identificando o operador.

tipo : símbolo predefinido indicando a posição e as regras de associatividade do operador, podendo ser : prefixa, sufixa, infixa associativa à esquerda ou à direita.

prior : número inteiro especificando, em relação a outros operadores, a ordem da avaliação do operador em uma expressão. A declaração efetuada através da cláusula "op" somente relata a existência do operador, sem definir sua semântica (o que o operador executa). Esta definição deve ocorrer através da inclusão no programa das cláusulas pertinentes ao operador.

A linguagem PROLOG possui um conjunto de operadores predefinidos e os mais importantes são: conjunção ",", disjunção ";" (infixo) e ":-" (infixo e prefixo).

O sistema PROLOG é construido de forma a possibilitar ao programador os mecanismos de programação das linguagens procedimentais, interface com o sistema operacional, etc.. As facilidades do sistema PROLOG são implementadas através de comandos conhecidos como predicado do sistema ou "built-in". Os predicados do sistema são inseridos nas cláusulas como se fossem átomos normais, porém atuam como interface para a função que executa a tarefa. Os comandos mais utilizados são os de manipulação aritmética, relacional, entrada e saída, e controle de execução.

No geral as implementações de PROLOG propiciam um conjunto suficiente de operadores aritméticos através de predicados predefinidos, tais como adição, produto e divisão. Em algumas implementações atuais encontramos predicados definidos para cálculo da raiz quadrada, coseno, seno, etc..

Os comandos para manipulacão relacional são: igual, diferente, maior ou igual, menor ou igual, maior e menor. Através destes comandos pode-se comparar números. Os operadores aritméticos e relacionais podem também ser utilizados para atribuição de resultados através da instânciação das variáveis que estejam como operandos.

Os comandos de entrada e saída permitem operar dados em arquivos ou periféricos. O PROLOG trabalha com o conceito de canais de entrada e saída de dados. Normalmente o canal de entrada é o teclado, e o de saída o vídeo. Entretanto, existem comandos que possibilitam alterar os canais de entrada e saída, e desta forma pode-se tornar por exemplo um arquivo texto no canal corrente de entrada. Existem comandos possibilitam a leitura tanto de simples caracteres quanto de dados formatados.

O sistema PROLOG possui comandos para criação de sua base de dados. O comando "consult" lê em um arquivo, um programa composto de cláusulas PROLOG e as cláusulas são inseridas na base de dados. O comando "reconsult" tem função semelhante a do "consult", porém elimina da base de dados todas as cláusulas com nome e numero de parâmetro igual as cláusulas contidas no arquivo lido.

Para utilizarmos o sistema PROLOG efetuamos uma sequência de passos que descreveremos a seguir. Primeiramente, através de um editor de textos, escreve-se um programa PROLOG. Em seguida entramos no ambiente PROLOG, e utilizando o comando "consult" carrega-se o programa para a base de dados PROLOG. Tendo-se feito esses passos pode-se então efetuar questões à base de dados.

O PROLOG possui comandos para: alterar, incluír e apagar cláusulas da base de dados durante uma sessão de consulta.

O PROLOG manipula cláusulas que são internamente representadas como listas. O sistema PROLOG possui predicados que transformam as listas em cláusulas e alguns outros predicados para a manipulação das listas.

Os comandos de controle "!" (cut), "true", "fail" e "repeat" são colocados à disposição, a fim de permitir aos programadores ferramentas para alterar a sequência natural efetuada pelo mecanismo de controle do PROLOG ao percorrer a árvore de busca. Estes comandos são importantes pois permitem a simulação de mecanismos de controle tais como "while", "for" e "repeat", que são largamente utilizados nas linguagens convencionais.

Buscou-se neste módulo dar uma visão ampla e sintética dos conceitos e características da linguagem PROLOG. Atualmente a literatura sobre a linguagem é bastante ampla, abrangendo desde livros que visam apenas a descrição da linguagem e que são voltados para os programadores, até livros que descrevem detalhes teóricos e modelos implementados, estes voltados aos pequisadores que desejam trabalhar no desenvolvimento da linguagem.

4.1.3 Implentações de lógica Nebulosa em PROLOG

A linguagem PROLOG tem sido largamente utilizada em trabalhos na Inteligência Artificial. Como a Inteligência Artificial visa criar sistemas computacionais para a modelagem do comportamento humano, é comum necessitarmos de mecanismos para manipular informações incompletas ou nebulosas. Nas aplicações da I.A. que utilizam o PROLOG e manipulam as informações incompletas através da lógica nebulosa, a solução frequentemente utilizada tem sido a construção de predicados PROLOG.

A seguir mostra-se dois modelos da implementação dos predicados PROLOG que operam a lógica nebulosa.

Os primeiros passos foram a construção dos predicados que obtinham os valores nebulosos nos fatos e no final efetuavam um cálculo nebuloso com os dados obtidos.

Por exemplo, para definir um predicado nebuloso "local_quente(Local,GRAU)", são fornecidos os dados sobre temperaturas médias e horas em que o sol está presente em várias localidades. O predicado é descrito a seguir com apenas os dados referente a uma localidade:

```
local_quente(Local,GRAU):-    temp(Local,T),quente(T,GRAU1),
                              sol(Local,S),claridade(S,GRAU2),
                              combine(GRAU1,GRAU2,GRAU).

temp(assis,28).
...
sol(assis,12).
...
quente(28, 0.7).
...
claridade(12, 0.6).
...
combine(Grau1,Grau2,Grau):- Grau1 > Grau2, Grau = Grau2
                                        ; Grau = Grau1.
```

Foram definidos quatro predicados para representar os fatos. O predicado "temp" que relaciona a localidade e sua temperatura média, o predicado "sol" que relaciona a localidade e a média de horas de sol, o predicado "quente" que representa um conjunto nebuloso quente que para cada valor da temperatura média possui associado um valor entre [0,1], e o predicado "claridade" que representa um conjunto nebuloso claridade que para cada valor da média de horas de sol possui associado um valor entre [0,1].

Portanto, se desejamos saber se a localidade Assis é quente efetuamos a seguinte consulta.

```
:-local_quente(assis,Grau), write(" Grau = ",Grau).
```

A cláusula "local_quente" é unificada e a variável "Local" é instanciada para "assis". Portanto passa-se a provar o corpo da cláusula. O predicado "temp(assis,T)" é unificado e a variável T é instânciada para 28; verificando a pertinência da temperatura 28 no conjunto nebuloso quente através do predicado "quente" obtemos na variável GRAU1 o valor 0.7. Da mesma forma, as cláusulas "sol" e "claridade" são resolvidas e a variável GRAU2 é instânciada com o valor 0.6. Tendo GRAU1 e GRAU2 sido instânciadas o predicado "combine" retornará na variável GRAU, o menor entre os dois valores, que neste caso é 0.6. O sistema Prolog retornará a seguinte mensagem:

```
Grau = 0.6
yes.
```

O resultado significa que o grau de pertinência da localidade Assis no conjunto nebuloso local quente é 0.6 .

No exemplo acima não foi colocado um valor para o limiar de aceitação do valor verdade. Colocando um limiar restringimos as respostas àquelas localidades que possuem valores maiores que o limiar. Isto significa que dado limiar=0.5, não podemos ter como resposta uma localidade com GRAU=0.3 . Conceitualmente estamos dizendo que uma solução acima do limiar estará contida no conjunto local quente e as soluções abaixo do limiar devem estar no conjunto local frio.

A seguir mostra-se uma outra forma da implementação do predicado "local_quente" onde: utiliza-se o conceito de limiar de aceitação, define-se operadores "e" e "ou" nebulosos (max e min) e não utiliza-se as variáveis GRAU1 e GRAU2.

```
local_quente(Local):-temp(Local,T), quente(T),
                     sol(Local,S), claridade(S),
                     f_and, fuzzy(GRAU),
                    write(Local, "é uma localidade quente com ",GRAU).


temp(assis,28).
...
sol(assis,12).
...
quente(28):-fuzzy(0.7).
...
claridade(12):-fuzzy(0.6).
...
```

A implementação foi efetuada utilizando uma estrutura similar a de uma pilha. Os valores de pertinência dos fatos nebulosos são colocados (empilhados) na base de dados e os operadores nebulosos recuperam (desempilham) os valores da base para efetuar cálculos sobre os mesmos e recolocam (empilham) o valor resultante na base.

Para inserir os fatos na base o PROLOG possui o predicado "assert" e para retirar o "retract". O valor do limiar corrente é colocado na base como parâmetro do fato "val_limiar" e neste exemplo assume o valor 0.5. O valor de pertinência dos fatos são colocados na base como parâmetros do fato "val_verdade".

A implementação acima difere da anterior na representação dos fatos nebulosos. Para representar os fatos foi criado o predicado auxiliar "fuzzy" cujo parâmetro é o valor do grau de pertinência associado ao fato. O predicado é usado para inserir o valor de pertinência na base se o mesmo é maior que o limiar, ou recuperar da base o valor verdade corrente.

```
fuzzy(Valor):- bound(Valor), asserta(val_verdade(Valor)),
                retract(val_limiar(Limiar)),
                asserta(val_limiar(Limiar)), !,
                Valor >= Limiar.
fuzzy(Valor):- retract(val_verdade(Valor)),
                asserta(val_verdade(Valor)), !.
```

A seguir estão as implementações dos operadores nebulosos. A cláusula "f_and" equivalente ao operador nebuloso "e", consiste em retirar dois valores da base e recolocar o menor deles. Da mesma forma a cláusula "f_or", equivalente ao operador nebuloso "ou", consiste em retirar dois valores da base e recolocar o maior deles. Podemos definir outros operadores desde que sigam idéia de recuperar os dados da base, operar sob os mesmos e recolocar o resultado a base.

```
f_and :- retract(truth(X)), retract(truth(Y)), !,
          fuzzy_min(X,Y,Z), fuzzy(Z), !.
f_or :- retract(truth(X)), retract(truth(Y)), !,
          fuzzy_max(X,Y,Z), fuzzy(Z), !.
fuzzy_max(X,Y,Z) :- X >= Y, Z = X; Z=Y.
fuzzy_min(X,Y,Z) :- X >= Y, Z = Y; Z=X.
```

Caso desejarmos efetuar novamente a consulta:

```
:- local_quente(assis).
```

serão colocados na base, pelas cláusulas nebulosas "quente" e "claridade", os valores 0.7 e 0.6 . A cláusula "f_and" recupera os valores da base e recoloca o menor deles . A cláusula "fuzzy" recupera o valor na variável GRAU. Assim teremos a resposta:

assis é uma localidade quente com 0.6

Nos exemplos dado acima é possivel perceber que os programas PROLOG que implementam a lógica nebulosa são difíceis de compreender e construir. Argumentos adicionais são inseridos nas cláusulas (valor de pertinência) e é necessario adicionar nos predicados um processamento apropriado para o cálculo (cálculo nebuloso).

Portanto, ao adicionamos lógica nebulosa a um dialeto PROLOG certamente perde-se significantemente o desempenho, pois inserimos uma nova camada entre o PROLOG e a aplicação.

Os resultados fornecidos ao usuário são difíceis de compreender pois parte dos predicados manipulam dados nebulosos e os demais mantêm-se como predicados convencionais.

Porém, o principal ponto a ser questionado nas implentações é que o "backtrack" não é feito sobre os resultados das operações nebulosas, pois os fatos nebulosos não são explicitamente parte da linguagem.

Pode-se também verificar que os operadores nebulosos utilizados são o mínimo e o máximo. Se forem efetuadas as implementações das propostas atuais dos operadores nebulosos os problemas destacados acima tenderão a aumentar e acentuarão a incapacidade do PROLOG em manipular as informações nebulosas.

Grande parte dos problemas citados acima, podem ser solucionado construindo-se um dialeto PROLOG que manipule a lógica nebulosa. Nas próximas seções discutiremos profundamente os conceitos do PROLOG e da lógica nebulosa para descrevermos e propormos modelos de PROLOG-nebuloso.

4.2 DEFINIÇÃO DOS MODÊLOS DE PROLOG NEBULOSOS

Nesta seção efetuamos a descrição teórica das linguagens de programação baseadas em lógica nebulosa e denominadas de PROLOG-nebuloso.

Inicialmente descreve-se um modelo de PROLOG-nebuloso denominado de convencional, pois possui as mesmas características do PROLOG convencional, e é baseado nos conceitos dos predicados nebulosos de primeira ordem e no princípio de resolução nebulosa definida por LEE [LEE72].

O segundo modelo descrito é denominado de Mukaidono. Possue algumas diferenças do modelo convencional, principalmente ao convencionar que todos os fatos unificam e nos aspectos lógicos utilizados. E, finalmente, o modelo aditivo onde discute-se os aspectos teóricos da implementação de um PROLOG-nebuloso que opera um cálculo aditivo que possibilita a implementação do raciocínio parcial e aproximado.

Na medida em que os modelos de PROLOG-nebulosos são apresentados verificaremos que cresce a capacidade de representação dos aspectos do raciocínio (manipulação de incerteza e raciocínio parcial e aproximado) e como consequência são realçadas as limitações e incompatibilidades com o PROLOG convencional e consequentemente com a lógica dos predicados.

4.2.1 MODELO CONVENCIONAL

Neste primeiro nível de modelagem, criaram-se sistemas que possuem a mesma estrutura básica e as características da linguagem PROLOG convencional. Portanto, o PROLOG-Nebuloso que iremos descrever é semelhante ao PROLOG convencional e manipula a lógica nebulosa, que neste contexto deve ser entendida como uma lógica de multivalores que suporta quase todas as definições da lógica clássica. Os operadores básicos, definidos em detalhes no capítulo 2, são: o "e" nebuloso é denotado por "&" e é equivalente a operação mínimo; "ou" nebuloso é denotado por "v" e é equivalente a operação máximo e o complemento nebuloso é denotado por "¬" (complemento). Este modelo de PROLOG-nebuloso foi implementado e a descrição da sua implementação está no proximo capítulo.

A linguagem PROLOG é entendida como a implementação da idéia de programação em lógica para um subconjunto de cláusulas definidas. Da mesma forma o PROLOG-nebuloso será descrito como a implementação da idéia de programação em lógica nebulosa para um subconjunto de cláusulas nebulosas definidas.

Um programa em PROLOG-nebuloso consiste de uma base de dados, contendo informações a respeito do assunto a ser questionado. A base de dados é montada por um conjunto de cláusulas nebulosas na forma das cláusulas de Horn.

Uma cláusula de Horn podem ser : uma cláusula definida que é uma cláusula do tipo "L :- M_1, M_2, ..., M_m" ou "L:-"; uma cláusula objetivo que é uma cláusula vazia ou do tipo "M_1, M_2,..., M_m".

Vimos anteriomente que a base de dados no PROLOG é composta somente de fatos e regras, que estão inteiramente associados ao assunto. Podemos então afirmar que o grau de pertinência dos fatos e das regras ao assunto é igual a 1. No PROLOG-nebuloso a base de dados é montada por fatos e predicados que estão associados ao assunto em diferentes níveis, portanto possuem um valor no intervalo [0,1], referente a esse grau de associação ou de pertinência ao assunto.

Portanto, quando definimos em PROLOG um predicado

gosta(eduardo,monica).

estamos efetuando uma afirmação de que sem dúvida alguma "Eduardo gosta de Monica". No PROLOG-nebuloso podemos expressar a intensidade (valor em [0,1]) da afirmação e rescrever a cláusula como:

gosta(eduardo,monica,0.8).

Neste modelo convenciona-se que as cláusulas que possuem valor de pertinência igual a 1.0 não necessitam ter o seu valor expresso, permitindo desta forma uma representação igual ao PROLOG.

Outro ponto importante a destacar é o valor verdade resultante de uma consulta. Para o PROLOG-nebuloso, assim como para algumas definições da lógica nebulosa, verdade nebulosa são valores contidos no intervalo (0.5, 1.0] . A adoção desta restrição está ligada a definição do princípio de resolução nebulosa que será visto a seguir. Entretanto este fato não impede que na base existam informações com valores nebulosos no intervalo [0.0, 0.5). Existem proposições mostrando que os especialistas utilizam frequentemente a lógica nebulosa positiva (valores (0.5, 1.0]) durante o seu processo de raciocínio. Isto pode ser visualizado quando manipulamos o conceito "alcoólatra". Sejam dadas as seguintes cláusulas:

```
alcoolatra(antonio, 0.8).
alcoolatra(jair,0.4).
```

representando os fatos "Antonio toma 3 copos de cachaça por dia" e "Jair eventualmente toma 1 copo de cachaça por dia". Considerando alcoólatra uma pessoa que toma um copo de cachaça por dia, nossa representação deve ser:

```
alcoolatra(antonio, 0.8).
nao_alcoolatra(jair, 0.6).
```

Embora o PROLOG-nebuloso permita representar informações com valores negativos (valores entre [0.0, 0.5)), estas situações não têm significado pois o princípio de resolução nebulosa opera apenas com as informações de valores positivos (valores entre (0.5, 1.0]). A representação das informações contidas no intervalo [0.0, 0.5) do domínio, devem ser feita criando-se um conjunto nebuloso que represente a negação da informação.

Caso desejarmos refinar as soluções podemos utilizar no PROLOG-nebuloso o conceito de limiar de aceitação do valor verdade nebuloso. O valor do limiar pode ser qualquer valor no intervalo (0.5, 1.0]. Inicialemente o PROLOG-nebuloso adota o valor do limiar igual a 0.5, entretanto se necessitarmos somente das soluções maiores que 0.8, basta apenas alterarmos o valor do limiar para 0.8 . Durante o processo de descrição do princípio de resolução exemplificaremos o conceito do limiar de aceitação do valor verdade nebulosa.

a) Valores Verdade Nebulosos e Princípio de Resolução

A obtenção do valor verdade no PROLOG-nebuloso é dividido em dois níveis. No primeiro nível o valor verdade de um predicado nebuloso depende do valor de cada uma das variáveis. Portanto seja C uma cláusula, denotaremos por T(C) o valor verdade nebuloso para uma dada interpretação da cláusula C. Então para um conjunto S de cláusulas, escrevemos similarmente o valor verdade nebuloso para uma dada interpretação como T(S), onde $S=C_1,C_2,..,C_n$, então T(S) significa

$$T(S) = T(C_1 \ \& \ C_2 \ \&...\& \ C_n) = \min(T(C_1),...,T(C_n)).$$

Portanto, consiste apenas em substituir as variáveis das cláusulas por valores nebulosos no intevalo (0.5, 1.0]. Obviamente o valor de T(S) está no intervalo (0.5, 1.0].

No segundo nível o valor verdade nebuloso de um predicado é obtido pelo valor das variáveis e através do sistema formal de resolução nebulosa.

As operações durante o processo de resolução no PROLOG são efetuadas sobre cláusulas que possuem grau de pertinência igual a 1.0. Para estendermos esse conceito criou-se o processo de resolução nebulosa que opera sobre as cláusulas que possuem grau de pertinência no intervalo (0.5, 1.0].

Para inferirmos informações de uma base de dados composta de cláusulas nebulosas criou-se um sistema formal de resolução nebulosa.

O sistema formal de resolução nebulosa trabalha exclusivamente com cláusulas nebulosas e contém apenas uma regra de inferência, a regra de resolução nebulosa, que gera uma nova cláusula a partir de duas outras.

Dado um conjunto S de cláusulas nebulosas e uma cláusula nebulosa C, uma dedução de C a partir de S neste sistema formal, consiste numa sequência de cláusulas terminando em C e geradas aplicando-se repetidamente a regra da resolução. Uma refutação a partir de S é a obtenção da cláusula vazia. Portanto, quando efetuamos uma consulta de uma cláusula nebulosa C ao conjunto S, necessitamos obter a cláusula vazia, ou seja, refutar a cláusula nebulosa C.

A regra da resolução nebulosa combina uma adaptação do Modus Ponnens e o processo de unificação.

A regra modus Ponens ou consequência lógica é dada por: dado A e $(\neg A \vee B)$ derivamos B. Esta representação pode também ser dada por A & $(\neg A \vee B)$. Então para valores nebulosos o resolvente é B com confiança "c" que é dada por $c = \max(T(A), T(\neg A))$

Seja S um conjunto de cláusulas. Denotaremos por R(S) o conjunto das cláusulas de S e todos os resolventes derivados de pares de cláusulas de S. Denotaremos por R^1 a primeira classe dos resolventes de S e sua confiança como c_1, então se após n passos $R^n(S)$ incluir somente a cláusula vazia, obteremos a refutação de S e o valor verdade nebuloso da confiança será dado por c^n=minimo(c^1,..,c^n). Sabendo-se que o domínio dos valores das variáveis é (0.5,1.0] podemos afirmar que o valor da confiança esta em :

$$\min [T(C_1),...,T(C_m)] \leq T(C^n) \leq \max[T(C_1),...,T(C_m)]$$

onde $C_1,...,C_m$ são cláusulas nebulosas de S e $T(C^n)$ é o valor verdade da confiança da cláusula gerada pela regra de resolução.

No exemplo abaixo utilizamos um conjunto de cláusulas para mostrar os passos da resolução nebulosa. Como nas cláusulas não existem variáveis o processo de resolução fica reduzido a aplicação do Modus Ponens. Portanto dado o seguinte conjunto de cláusulas:

1. $A \vee \neg B$
2. $B \vee \neg D$
3. D

Ao efetuarmos a consulta dada na cláusula 4 esta é inserida ao conjunto de regras então no exemplo R(S) é dado pelas regras numeradas de 1 a 4. As regras de 5 a 7 são os resolventes gerados pela regra da resolução

4. $\neg A$
5. $\neg B$ gera R^1 a partir de 1 e 4 , e c^1= max(T(A),T($\neg$A))
6. $\neg D$ gera R^2 a partir de 2 e 5, e c^2= max(T(B),T($\neg$B))
7. $\square$ gera R^3 a partir de 3 e 6, e c^3= max(T(D),T($\neg$D))

O valor da confiança ou valor verdade nebuloso obtido na consulta é dado por:

$$c^4 = \min (c^1, c^2, c^3).$$

Para obtermos a solução basta atribuirmos valores nebulosos para A, B e D e desta forma obtermos o valor verdade nebuloso.

O passo da unificação é o segundo componente da regra de resolução.

O passo da unificação é aplicado quando desejamos tornar idênticos literais de um conjunto de cláusulas através da substituição de variáveis por termos. As substituições podem ser efetuadas em dois níveis. O primeiro consiste em tornar idênticos literais complementares oriundos de cláusulas diferentes, como ocorre na geração da cláusula 5 do exemplo abaixo, ao substituir (X/lula, Y/caiado) na cláusula 1. O segundo nível consiste em tornar idênticos literais de uma mesma cláusula.

As propriedades da resolução nebulosa asseguram que a ordem na qual os literais são cancelados não influência o resultado final da prova. Entretanto, o número de passos até o objetivo é dependente dos literais escolhidos na substituição.

Abaixo é exemplificado a aplicação da regra da resolução nebulosa em um conjunto de regras. Dado as regras abaixo:

1. amigos(X,Y) ∨ ¬afinidade(X,Y,K) ∨ ¬afinidade(Y,X,L).
2. afinidade(lula,caiado,0.7).
3. afinidade(caiado,lula,0.6).

Ao efetuarmos a consulta da cláusula 4 geramos, pela regra de resolução as cláusulas subsequentes:

4. ¬amigos(lula,caiado). /* consulta
5. ¬afinidade(lula, caiado, K) ∨ ¬afinidade(caiado, lula,L). /* gerada a partir de 1 e 4 e as substituições (X/lula,Y/caiado).
6. ¬afinidade(caiado, lula, 0.6) /* a partir de 2 e 5
7. □ /* a partir de 3 e 6

As cláusulas nebulosas com o grau de pertinência igual a 1.0, não possuem seu valor representado. Portanto o valor da confiança ou valor verdade nebuloso é obtido por

$$c^7 = \text{minimo}(c^4, c^5, c^6) = 0.6$$

e o domínio das possíveis soluções é dado por $0.6 \leq T(C^n) \leq 1.0$ para n=5,..,6.

b) Estratégia do Prolog Nebuloso

Podemos representar todos os possíveis caminhos gerados pela aplicação da regra de resolução para um conjunto de cláusulas através de uma árvore de busca.

A estratégia usada para implementação do PROLOG-nebuloso consiste em percorrer uma árvore de busca em profundidade a procura da primeira solução. A busca de solução é restrita às cláusulas com grau pertinência em (0.5, 1.0], o que equivale a trabalhar com o conceito de limiar de aceitação com valor igual a 0.5. A seguir descreveremos como é montada a estratégia do PROLOG-nebuloso

A consulta a uma base de dados nebulosa é feita por uma cláusula objetivo nebulosa, contendo um conjunto de procedimentos, como vemos abaixo:

$$:- A_1, A_2, \ldots, A_n.$$

O sistema PROLOG-nebuloso buscará resolver todos os procedimentos da consulta da seguinte forma: toma-se a cláusula consulta e invoca-se o primeiro sub-objetivo nebuloso A_1. Pesquisa-se na base de dados uma cláusula A_1, com mesmo nome, número de parâmetros e com grau de pertinência maior que o limiar de aceitação (que nesta situação é igual a 0.5). Se todas as condicões forem satisfeita a unificação sucederá e se existirem variáveis serão efetuadas as substituíções apropriadas. Então será criada uma nova cláusula objetivo, substituindo o literal A_1 pelo corpo da cláusula unificada.

Seja

$$A_1 :- B_1,B_2,...,B_m.$$

a cláusula da base que foi unificada, a cláusula objetivo é alterada e passa a ser:

$$:- (B_1,B_2,...,B_m,A_2,...,A_n)\theta$$

onde θ representa as substituições mais simples possíveis das variáveis por termos do sub-objetivo A_1 e da cláusula A_1 contida na base de dados.

A consulta sucederá quando todos os procedimentos contidos na lista de objetivos forem executados (ou, através do processo de resolução, obtenha-se a lista vazia).

Quando um sub-objetivo, por exemplo B_1, não unificar, por não existir uma cláusula nebulosa B_1 correspondente na base de dados, ou porque o valor do grau de pertinência de B_1 é menor que o limiar de aceitação (o sistema assume 0.5), o sistema nebuloso retorna (também chamado de "backtrack") à última unificação, que neste caso de B_1 é A_1. Este processo é acompanhado por descartar as substituíções θ criadas pela última unificação A_1, e A_1 é novamente o subobjetivo a ser buscado na base, e esta pesquisa é feita descartando as cláusulas já unificadas por A_1. Caso o "backtrack" ocorra e não exista um ponto de retorno, em que se possa obter um novo sub-objetivo, a consulta falha.

c) Programas e Predicados do PROLOG-Nebuloso

Programar em PROLOG-nebuloso consiste em montar uma base composta por fatos e regras nebulosas sobre o escopo de um dado problema. O sistema PROLOG-nebuloso é igual ao sistema PROLOG convencional e deve possuir mecanismo para interpretar os fatos e as regras do programa, montar uma base de dados na área de trabalho do PROLOG-nebuloso e possibilitar a execução de inferência sobre estes fatos através da processo de resolução e dos mecanismos de controle de estratégia.

Os fatos ou predicados nebulosos são representados na forma de uma cláusula unitária básica

$$p(x_1,x_2,...,x_n,\mu)$$

onde "p" é o simbolo do predicado, e x_i para i=1..n podem ser variáveis individuais e frequentemente independente ou serem constantes (atomos), e μ representa o grau de pertinência ou grau de associação do relacionamento que em algumas situações é ausente. O predicado acima é dito ter "n" parâmetros.

Estas cláusulas unitárias são utilizadas para representar fatos referentes ao universo do discurso, pois dentro da teoria dos conjuntos nebulosos, fatos correspondem à representação de elementos do conjunto. Os fatos

```
alto(1.80, 0.8).
pai(joao, miguel).
```

são fatos nebulosos e aqueles em que o grau de pertinência é igual a 1, como no fato "pai(joao, miguel).", não necessitam ter seu valor representado. Pode-se também representar elementos de relações nebulosas como a relação "gordo" na qual relaciona-se altura com peso:

```
gordo(170, 120, 0.9).
```

e leia-se "uma pessoa com 170 cm e 120 quilos possui 0.9 de pertinência no conjunto gordo"

Os predicados nebulosos descritos acima estão ligados à representação de fatos associados a conjuntos e relações nebulosas enumeráveis. No entanto o sistema PROLOG-nebuloso, assim como alguns sistemas PROLOG possuem um conjunto de predicados, os quais são conhecidos como "built-in". Estes predicados são fornecidos aos usuários a fim de permitir funções extra lógicas, como, por exemplo, predicados de entrada e saída ou predicados para manipulação de estruturas que são básicas no PROLOG como a estrutura de dados. Dentro destes conceitos um PROLOG-nebuloso deve conter um conjunto de predicados que implemente as funções matemáticas nebulosas mais comuns e desta forma permita a representação de conjuntos nebulosos contínuos e variáveis linguísticas nebulosas. Quando associamos um fato a uma função matemática continua estamos garantindo que para qualquer valor o fato unifica.

Outra forma de representação no PROLOG-nebuloso convencional são as regras, representadas na forma de cláusulas não unitárias e que expressam as relações existentes no domínio do problema. As regras no PROLOG-nebuloso convencional não diferem das regras do PROLOG e são dadas da seguinte forma:

```
funcionario_ideal(X):- desempenho_mensal(X,K),
                       assiduidade(X,K),
                       salario(X,K).
```

e leia-se "funcionário ideal é aquele que possue um bom desempenho mensal, assiduidade e faixa aceitável de salário". Os predicados "desempenho_mensal", "assiduidade" e "salario" são predicados nebulosos.

Uma consulta à base de dados é feita utilizando-se uma cláusula objetivo, como por exemplo:

```
:-funcionario_ideal(antonio).
```

d) Implementação do Limiar de Aceitação

Uma característica do PROLOG-nebuloso convencional é que o processo unificação leva em consideração o limiar de aceitação. Desta forma uma cláusula deve possuir o valor do grau de pertinência superior ao limiar de aceitação para poder sofrer o processo de unificação.

O processo de obtenção do valor verdade nebuloso é efetuado durante os passos da resolução. Ao iniciarmos uma consulta o valor verdade nebuloso do sistema recebe o valor 1.0 . Após o processo de unificação ter sucedido efetua-se uma comparação entre o valor verdade nebuloso do sistema e o valor da confiança (este processo equivale à obtenção do grau de pertinência da cláusula eliminada pelo processo de resolução). Após esta comparação, o valor verdade nebuloso do sistema deverá conter o menor dos valores. Desta forma ao final da consulta teremos no valor verdade nebuloso do sistema o valor nebuloso final da consulta, que equivale ao menor valor das confianças geradas durante a resolução.

Podemos alterar o valor do limiar de aceitação para valores no intervalo (0.5, 1.0], e desta forma refinar as soluções obtidas. Se tomarmos o limiar de aceitação com valor igual a 1.0 teremos novamente o PROLOG convencional. O uso do conceito do limiar de aceitação é na realidade a implementação do conceito de conjunto nebuloso de nível α.

Para exemplificar os passos de uma consulta e os cálculos de um PROLOG-nebuloso construiremos a seguir um programa simples na sintaxe do PROLOG-10 que é a mesma do interpretador PROLOG-nebuloso PRONEB que foi implementado e será descrito no proximo capítulo.

```
novato(Pessoa) :- idade(Pessoa,Y), novo(Y,K).
idade(maria,30).
idade(glauco,28).
idade(adriana,23).
novo(30, 0.6).
novo(28, 0.7).
novo(24, 0.9).
novo(23, 0.95).
novo(20, 1.00).
```

O predicado "novo" representa um conjunto nebuloso, e o valor do limiar de aceitação, neste exemplo, é igual a 0.5 . Ao efetuarmos a seguinte consulta:

```
:-novato(glauco).
```

o PROLOG nebuloso efetua os seguintes passos: A variável que acumulará o valor verdade nebulosa durante a resolução é inicializada com 1.0. Toma-se a cláusula objetivo "novato(glauco)" e a seguir verifica se na base existe um predicado correspondente. Se a busca obtiver sucesso e os predicados envolvidos possuem valor do grau de confiança maior que o limiar, consequentemente a unificação é executada e a substituição (Pessoa/glauco) é efetuada e o valor verdade nebuloso não é alterado pois o valor da confiança dos predicados é 1.0 (assume-se valor de pertinência igual a 1.0 quando os mesmos não forem representados). Uma nova lista de objetivo é gerada substituindo o predicado unificado pelo corpo da regra contida na base com todas as substituições efetuadas:

```
:-idade(glauco,Y), novo(Y,K).
```

Agora o primeiro sub-objetivo da lista ("idade(glauco,Y)") é tomado para continuar o processo de resolução. A pesquisa na base é efetuada, e a unificação ocorre com a substuição (Y/28). O valor verdade nebuloso mantém-se, e a lista de objetivo reduz-se a:

```
:-novo(28,K).
```

A unificação de "novo(28,K)" com o fato "novo(28,0.7)" é efetuada. Como o valor verdade nebuloso do sistema é maior que a confiança, o mesmo é atualizado e recebe o valor da confiança 0.7. A lista da consulta torna-se vazia, portanto a consulta sucedeu e a seguinte mensagem é dada

```
verdade 0.7.
```

Se o limiar de aceitação for alterado para o valor 0.8 e quando efetuamos a mesma consulta acima, o PROLOG-nebuloso não unificará as cláusulas "novo(28,K)" e "novo(28, 0.7)", pois o valor da confiança 0.7 é menor que o limiar 0.8. Uma nova busca é efetuada na base e a cláusula "novo(25,K)" não unifica com as cláusulas da

base. Portanto a consulta falha e é impresso a seguinte mensagem:

falso 0.0.

e) Mecanismos Extra-Lógicos

Podemos implementar no PROLOG-nebuloso convencional mecanismos extra lógicos tais como o "cut", "repeat" e outros. Portanto o PROLOG-nebuloso convencional pode proporcionar aos programadores mecanismos para melhorar a eficiência e permitir o paradigma procedural.

O sistema PROLOG-nebuloso para tornar-se efetivamente útil deverá conter um conjunto de predicados ("built in") que permitam ao programador, entre outras coisas, facilidades na construção de programas, montagem da base de dados e interface com o sistema operacional. Pode-se afirmar que exceto as diferenças destacadas anteriormente, um sistema PROLOG-nebuloso convencional pode conter todas as características do PROLOG descritas na seção 4.2.2 .

Com o crescimento da popularidade do PROLOG como linguagem de programação um grande número de profissionais foram treinados para utiliza-la e paralelamente um grande número de software reutilizáveis foram gerados. A definicão do PROLOG-nebuloso convencional, permite o reaproveitamento de profissionais e softwares escritos em PROLOG.

Apesar das vantagem demonstradas acima, veremos nas definições dos próximos modelos, algumas críticas e desvantagens que tornam este modelo impróprio para manipular lógica nebulosa e consequentemente formalizar o raciocínio de especialistas.

4.2.2 MODÊLO MUKAIDONO

Este modelo de PROLOG-nebuloso, foi implementado no laboratório Mukaidono da Universidade de Meije [MUKA87 et alli], e resulta de estudos que o grupo tem feito sobre lógica nebulosa e princípio de resolução nebulosa, a fim de permitir ao PROLOG-nebuloso uma estratégia de inferência eficiente.

Embora baseado nos operadores nebulosos máximo e mínimo, este modelo se diferencia do anterior pois possui agregado ao processo de resolução um novo modo de obtenção do valor verdade nebuloso.

O modo de obtenção do valor verdade nebuloso é diferenciado em três níveis. No primeiro, o cálculo do valor verdade de um predicado depende do valor atribuído a cada variável; no segundo, é utilizado o peso da regra que consiste do relacionamento lógico do valor verdade nebuloso da premissa e o valor verdade da conclusão, e finalmente no terceiro nível o cálculo é obtido através das confianças geradas pelo princípio de resolução nebulosa durante uma consulta.

Portanto, este modelo efetuará basicamente os mesmos passos do anterior. Entretanto, possui outras formas de operação sobre os valores gerados durante o processo de resolução.

O modelo anterior é fundamentado no princípio de resolução nebulosa definida em [LEE72], onde o princípio de resolução nebulosa opera valores nebulosos no intervalo (0.5, 1.0]. Neste modelo o princípio de resolução foi definido ser significativo para valores no intervalo [0.0, 1.0]. Portanto as variáveis nebulosas podem receber valores em [0.0, 1.0].

As definições dos conectores lógicos ("&", "v") e suas respectivas operações (mínimo e máximo) mantêm-se iguais as definições anteriores da lógica nebulosa.

a) Níveis de Interpretação

O sistema permite três modos para obtenção do valor verdade nebuloso de cláusulas nebulosas.

No primeiro, também encontrado nas implementações anteriores, o valor verdade nebuloso de uma cláusula C é determinado unicamente pela substituição ou mapeamento de valores [0.0, 1.0] nas variáveis da cláusula, e é denotada por T(C). Para um conjunto de cláusulas $S=(C_1,C_2,..,C_n)$ o valor verdade nebuloso, denotado por T(S), para uma dada interpretação, é obtido por

$$T(S) = T(C_1 \& C_2 \& ... \& C_n) = \min(T(C_1),...,T(C_n))$$

O segundo modo de obtenção do valor verdade nebuloso é o princípio de resolução nebulosa. Vimos no modelo anterior que a regra de resolução é baseada no modus Ponens, e neste modelo dado A e ($\neg A \vee B$) o resolvente obtido é B, e o cálculo da confiança é o valor absoluto da confiança de A e é dado por:

$$c = | c_A | = ((\max(T(A), T(\neg A)) - 0.5) * 2.$$

O valor gerado no cálculo da confiança de A (c_A) está no intervalo [-1, 1]. Ao aplicarmos o módulo obtemos novamente um valor em [0,1]. Dado um conjunto S de cláusulas nebulosas, ao efetuarmos uma consulta (C) sobre S e se após *n* passos do processo de resolução obtivermos a cláusula vazia, então o valor verdade nebuloso da consulta (c) é dado pelo mínimo valor das *n* confianças dos resolventes gerados pelo

processo de resolução ($c = \min(c_1,..,c_n)$).

Uma das vantagens deste cálculo da confiança ((max(T(¬A),T(A)) - 0.5) * 2) é que podemos eliminar o problema da ambiguidade (valor próximo a 0.5), pois o valor da confiança tende a ser mais próximo aos extremos (0 e 1) e permite-nos afirmar que a consequência lógica gerada é menos ambígua que a premissa e assim pode-se trabalhar com valores nebulosos em [0,1].

A regra da resolução deste modelo difere do anterior no cálculo da confiança (c) e como consequência toma-se um novo domínio de valores nebulosos ([0,1]). As demais definições são as mesmas efetuadas no modelo anterior.

O terceiro modo é o conceito de peso que não possui similar na implementação anterior. O peso é um valor que representa o relacionamento lógico entre o valor da confiança da premissa e o valor da confiança da conclusão.

Seja uma regra A → B, sabemos que T(A→B) é dado por T(¬A)∨T(B) (equivalente a max(T(¬A),T(B)). Entretanto, se tivermos T(A) (T(B)) e T(A→B) não podemos obter o valor de T(B) (ou T(A)) quando T(A→B) = T(¬A) (ou T(A→B) = T(B)). Isto porque o operador "∨" obtém o maior dos valores, e não se pode determinar o menor valor que é dado por T(B) (ou T(A)). Para solucionar criou-se um relacionamento semântico entre premissa e conclusão o qual denominamos de peso.

Seja P→Q uma regra, T(P) e T(Q) ∈ [0,1] são os valores verdade nebuloso da premissa e da conclusão respectivamente. Então o peso da regra $W_{P\to Q} \in [-1,1]$ é dado por $W_{P\to Q} = C_Q * C_P$ onde "*" significa o operador produto e C_Q e C_P são as confianças que são dadas por $C_P = (T(P) - 0.5) * 2$ e $C_Q = (T(Q) - 0.5) * 2$.

Se $W_{P\to Q} > 0$ então a premissa e a conclusão estão no mesmo intervalo [0.0, 0.5] ou (0.5, 1.0], caso $W_{P\to Q} < 0$ então a premissa e a conclusão estão em intervalos diferentes. Portanto, tendo o valor da confiança da premissa e o valor do peso da regra pode-se obter o valor da confiança da conclusão.

Embora a definição do valor dado ao peso dependa do programador programador do sistema, este valor não deve ser aleatório pois é fundamental para o cálculo. A forma correta da obtenção deste valor, seria durante a fase de aquisição de conhecimento tal como é efetuada na obtenção do valor da adesão utilizada no cálculo aditivo descrito no capítulo 2.

Além da informação da confiança resultante da consulta, este modelo apresenta como solução o valor de um cálculo nebuloso chamado de mistura.

Seja uma consequência C cuja confiança é denotada por c_c e a confiança dos resolventes denotada por c, então o valor verdade nebuloso da mistura da consequência C é dado por

$$MT(C) = cr(C) * 0.5 + 0.5 \quad \text{onde } cr(C) = c_c * c \text{ e } MT(C) \in [0,1]$$

O mecanismo de inferência é basicamente o mesmo utilizado no PROLOG-nebuloso convencional. Entretanto, este modêlo não trabalha com o conceito de limiar e os vários cálculos descritos acima, são especificos deste modelo e são executados durante o processo de resolução.

b) Unificação Nebulosa e Predicados Nebulosos

Vimos anteriomente na descrição do PROLOG e do PROLOG-nebuloso convencional, que ao tentarmos unificar um fato com a base de dados o processo pode suceder ou não. De certa forma nestes casos a unificação também funciona como na lógica clássica (não unifica, unifica). Partindo deste princípio podemos afirmar que em um PROLOG-nebuloso a unificação de um fato com a base sempre deve ocorrer, ou seja, uma unificação nebulosa deve retornar um valor no intervalo [0,1] representando a unificação do fato com a base.

Neste modelo foram implementadas formas de unificação nebulosa através da definição de predicados nebulosos. Um predicado nebuloso pode ser considerado uma função matemática cujos parâmentros são elementos no universo do discurso contínuo e que retorna o grau de pertinência que é um valor em [0,1]. Um predicado nebuloso é a implementação dos conceitos de conjuntos nebulosos, relações nebulosas e variáveis linguísticas implementadas no capítulo 3.

Por exemplo podemos criar neste modelo um predicado nebuloso "jovem(X)", onde X é uma variável, que representa um conjunto nebuloso "jovem" e possui associado uma função matemática como a descrita a seguir:

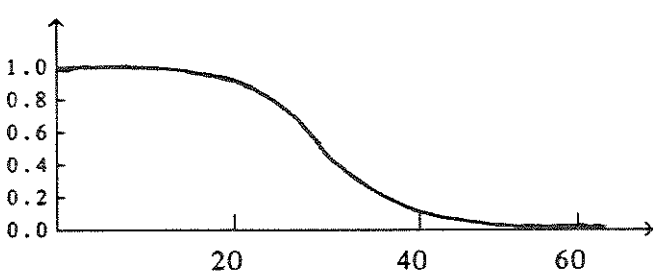


Fig. 4.4: Função matemática correspondente ao predicado "jovem"

Para qualquer valor atribuido a variável X, que esteja contido universo do discurso, o predicado "jovem" irá unificar e retornar um valor nebuloso.

O predicado "jovem" acima é um conjunto contínuo. Entretanto é comum a existência de dados discretos e sem funções associadas. Nestas situações existem ao menos duas soluções.

A primeira, que está implementada neste modelo, permite ao processo de unificação gerar, a partir dos fatos contidos na base, um valor aproximado através da interpolação. Por exemplo seja uma base:

```
...
jovem(20, 0.9).
jovem(30, 0.7).
...
```

ao efetuarmos uma consulta "jovem(25, K)" o processo de unificação gerará através de interpolação o valor 0.8.

O segunda forma, que é apenas filosófica e ainda impossível de implementar, consiste em permitir ao processo de unificação mecanismos para trabalhar com o conceito de similaridade. Dado um fato busca-se na base um fato correspondente para unificar. Caso não exista uma unificação simples, procura-se um fato similar na base ou de acordo com os fatos existentes na base gera-se um valor nebuloso aproximado. Este conceito é a implementação dos mecanismos utilizados pelos especialistas ao depararem com um fato novo. Certamente ele não pode associar um valor correto àquele fato, porém, devido a sua experiência anterior (base de dados), ele associa algum valor ao fato. Em situações de desconhecimento total o especialista associa o valor 0.5 para indicar que ele não pode afirmar nada sobre o fato (atribuindo valores acima de 0.5) ou negar (atribuindo valores menores que 0.5).

Para melhor compreender todos os passos dos cálculos efetuados por este modêlo, durante uma consulta veja o exemplo abaixo.

Seja um programa escrito na sintaxe definida por este modêlo:

```
juventude(Pessoa) :- idade(Pessoa, Idade), jovem(Idade) 0.5
idade(Pessoa, Idade) :- 0.9/('joao',17), 0.8/('antonio',25)
                                                repeat #2.
jovem(Idade) :- 1.0/(17), 0.75/(25), 0.6/(30) repeat #3
```

O valor 0.5 na associado a regra "juventude" é o peso da regra. A sintaxe permite representarmos fatos de um conjunto nebuloso de modo sintético utilizando o "repeat" como visto em "idade" e "jovem".

Ao efetuarmos a seguinte consulta

?-juventude(antonio).

o sistema efetuará os seguintes cálculos. O primeiro passo é o cálculo da verdade da premissa

$$T(premissa)=min(T(idade(antonio,25)),T(jovem(25))$$
$$=min(0.8,\ 0.75) = 0.75$$

então podemos calcular a confiança da premissa por

$$c_{premissa} = (T(premissa)-0.5) * 2 = (0.75-0.5) * 2 = 0.5.$$

Sendo o peso w=0.5 o valor da confiança da conclusão é dado por

$$c_{juventude} = w / c_{premissa} = 0.5 / 0.5 = 1.0$$

O valor verdade nebuloso da conclusão pode ser obtido por

$$T(juventude(antonio)) = c_{juventude} * 0.5 + 0.5 =$$
$$= 1.0 * 0.5 + 0.5 = 1.0.$$

O valor da confiança dos resolventes é dado por

$$c = min\ (|\ c_{premissa}\ |, |c_{juventude}\ |) = min\ (0.5,\ 1.0) = 0.5.$$

Sendo a confiança do resolvente consequência dada por

$$cr(juventude(antonio)) = c_{juventude} * c = 1.0 * 0.5 = 0.5$$

podemos então calcular o valor verdade nebuloso mistura da consequência "juventude(antonio)" por

$$MT(juventude(antonio)) = cr(juventude(antonio)) * 0.5 + 0.5$$
$$= 0.5 * 0.5 + 0.5 = 0.75.$$

Ao final do cálculo o sistema nebuloso colocará a seguinte mensagem

(MT = 0.75 T = 1.0 CONF = 0.5)

que significa que "Antonio está em plena junventude (valor verdade da conclusão T=1.0) e a afirmação é 50% verdadeira (confiança dos resolventes CONF=0.5)". O valor de MT representa o nível de ambiguidade da resposta (quanto mais distante de 0.5 menor é a ambiguidade).

4.2.3 MODELO ADITIVO

Vimos que a lógica é a ciência que estuda as leis do raciocínio. Portanto, uma linguagem de programação em lógica deve possuir todos os mecanismos utilizados durante o raciocínio. Dentro deste conceito o PROLOG (linguagem de programação em lógica) deve possuir mecanismos para manusear informações imprecisas e inferir sobre as mesmas, efetuar raciocínio com dados parciais (situação onde não é necessário obter todas as informações a respeito de uma hipótese para que a mesma seja aceita) e o

raciocínio aproximado (situação onde na medida em que as informações são fornecidas elas são agregadas gerando símbolos com confiança maior ou igual à confiança dos dados).

Analisando o PROLOG tradicional, verificamos que trata-se de uma ferramenta que representa o conhecimento completo através de cláusulas e o processo da resolução opera sobre as mesmas para inferir informações. O mecanismo para busca de soluções é feito em profundidade, onde se obtem a primeira solução.

Podemos, como vimos na descrição dos modelos de PROLOG-nebulosos, criar uma linguagem de programação em lógica capaz de representar as informações completas e as incompletas, e com um mecanismo de inferência em lógica nebulosa. Portanto, um PROLOG-nebuloso cobre parte dos mecanismos manipulados pelo raciocínio.

Os cálculos utilizados nos modelos de PROLOG-nebulosos descritos estão muito ligados ao cálculo utilizado na lógica clássica. Como consequência, existe todo um desenvolvimento teórico efetuado durante anos, que suporta sua utilização. Enquanto os cálculos dos modelos descritos foram criados baseados em aspectos teóricos, a proposta do cálculo aditivo segue um caminho oposto. O cálculo aditivo foi concebido ao verificar-se que os dados obtidos durante as entrevistas de aquisição de conhecimento com os especialistas, não encaixavam-se em nenhum dos modelos existentes.

O operador aditivo e/ou que é descrito no capítulo 2, representa uma das tendências nas pesquisas de operadores na lógica nebulosa. Através de um único operador nebuloso, que configurado pode ter o comportamento de operadores da lógica clássica, de operadores tradicionais da lógica nebulosa (min-max) e um comportamento aditivo (que pode ser utilizado para raciocínio aproximado e raciocínio parcial).

O operador aditivo formulado abaixo:

$$c(H) = (1-\alpha)* \sqrt[p]{\prod_n a(f_i)*c(f_i)} + \alpha*(1- \sqrt[p]{\prod_n (1-a(f_i)*c(f_i))})$$

utiliza quatro parâmetros, a confiança "c", a adesão "a", um valor no intervalo [0,1], denominado de α, que determina o comportamento do operador (clássico, nebuloso).O parâmetro "n" é o número de parâmetros, que na sintaxe PROLOG representa o número de cláusulas no corpo da regra e "p" é dado pelo usuário. Se "p" é configurado com valor menor que o número de cláusulas ("n") o cálculo terá comportamento aditivo, e se tiver valor maior seu comportamento será restritivo.

Nas entrevistas com os especialistas os dados são obtidos, através de uma metodologia de aquisição de conhecimento (Método MAR [LEAO90]), que os representa na forma de grafos nebulosos. Tem-se em uma única estrutura (os grafos), os dados e as sequências em que os dados são agregados para efetuar a prova de uma hipótese. De certa forma o grafo representa a sequência do raciocínio efetuada pelo especialista.

As informações que compõem a base de dados são padrões fornecidos pelos especialistas (grafos e conjuntos nebulosos). Os parâmetros do operador (adesão, confiança e os dados da base) são ponderados dentro do contexto do problema. Consequentemente, o cálculo sobre estes parâmetros deve ser mais consistente.

Nesta seção efetuaremos a discussão do operador aditivo no contexto da estrutura PROLOG. Ao contrário dos modelos anteriores, em que temos a discussão teórica e existem protótipos implementados (no proximo capítulo descreve-se a implementação de um modelo de PROLOG-nebuloso convencional (PRONEB) e o modelo PROLOG-Mukaidono pode ser visto em [MUKA87 et alli]), a descrição deste modelo restringirá apenas as discussões teóricas pois este modelo será implementado.

Abaixo revemos o mesmo grafo apresentado no capítulo 2, obtido durante uma sessão de aquisição de conhecimento.

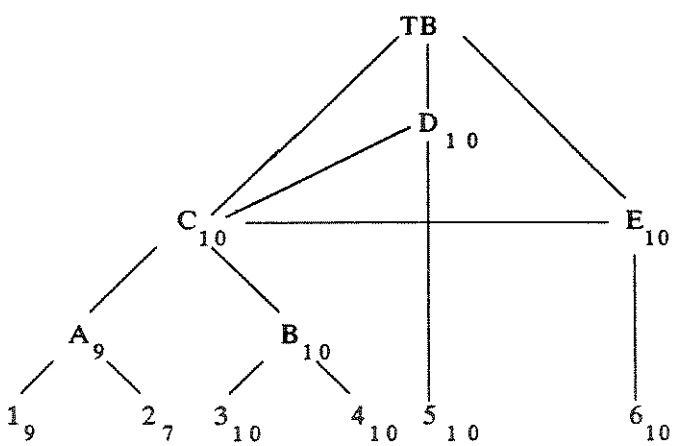


Fig. 4.5 : Grafo de conhecimento sobre tuberculose

Os fatos dos nodos inferiores estão representados por números e representam:

1 - Tosse por mais de duas semanas

2 - Febre

3 - Raio X

4 - Presença de Bacilus no Catarro

5 - Presença de Bacilus na Cultura

6 - Biópsia sugestiva

A representação do grafo em uma sintaxe semelhate à dos PROLOG-nebulosos descritos seria

tuberculose(Pessoa) :- c(Pessoa). <10>(0,1)

tuberculose(Pessoa) :- c(Pessoa) <10>, bacilus_cultura(Pessoa) <10>. (0,2)

tuberculose(Pessoa) :- c(Pessoa)<10>, biopsia(Pessoa) <10>. (0,2)

c(Pessoa):-a(Pessoa)<9>, b(Pessoa)<10>.(0,1)

a(Pessoa):-tosse(Pessoa) <9>, febre(Pessoa)<7>.(0,1)

b(Pessoa):-raio_x(Pessoa) <10>, bacilus_catarro(Pessoa)<10>. (0,2)

Na sintaxe acima podemos verificar as primeiras diferenças do conteúdo das regras dos modelos dos PROLOG-nebulosos anteriores e das regras acima. Os valores entre "< >" após as cláusulas, representam a adesão, e em cada regra temos entre parênteses, os valores fornecidos pelos programador para configurar o operador (aditivo, nebuloso,..). O primeiro parâmetro de configuração é o alfa e o segundo o número de cláusulas.

a) Obtenção do valor da Confiança e Resolução

O cálculo da confiança é dividido em dois níveis de interpretação. O primeiro nível, que é comum a todos os modelos descritos, consiste em substituir na fórmula do operador osparâmetros e obtermos o valor da confiança.

No segundo nível, a confiança da regra é obtida por substituir na fórmula os parâmetros fornecidos na regra (adesão, alfa e número de cláusulas) e os valores das confianças dos resolventes gerados durante o processo de resolução.

O processo da resolução é baseado numa adaptação do Modus Ponens, que é dado por A & ($\neg$A $\vee$ B), onde o resolvente é B e a confiança (c_A) é dada pelo aplicação do operador aditivo e/ou sobre os valores de $\neg$A e A . Nos outros modelos o operador utilizado era o operador nebuloso máximo; neste modelo podemos utilizar o operador aditivo e/ou com comportamento por exemplo: aditivo ou restritivo. Para um conjunto S de cláusulas nebulosas, ao efetuarmos uma consulta (C) sobre S e se após *n* passos do processo de resolução obtivermos a cláusula vazia, o valor verdade nebuloso da consulta será dado pela aplicação do operador aditivo e/ou a todos os *n* valores das confianças dos resolventes gerados pelo processo de resolução ($c=F(c_1,c_2,..,c_n$). Nos modelos anteriores F corresponde ao operador mínimo.

Para garantirmos a validade do processo de resolução necessitamos efetuar algumas restrições ao comportamento de A, pois se o operador estiver configurado para o cálculo aditivo pode-se gerar a confiança com valor maior que 1.

Portanto o comportamento de A deve ser modulado tal como a função abaixo.

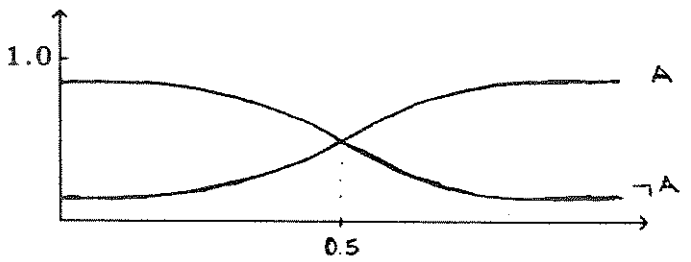


Fig 4.6: Comportamento da A

Pode-se ver na figura acima, que A possui o comportamento de uma sigmoide que não necessita ser complementar, bastando apenas obedecer a ¬A + A < 1 e seu comportamento deve ser simétrico. Desta forma pode-se garantir que mesmo o operador estando configurado para o comportamento aditivo, o valor calculado da confiança não irá ultrapassar a 1.

A definição do gráfico acima (Fig 4.6) não se opõe aos conceitos da lógica nebulosa e aos conceitos manipulados pelos especialistas. Durante as entrevistas de aquisição do conhecimento pode-se verificar que os especialistas não trabalham a complementariedade e o ponto de ambiguidade é o valor 0.5. Quando um especialista não sabe a respeito de um dado ele atribui àquele dado o valor 0.5. Este comportamento também pode ser visto no gráfico, pois é no valor 0.5 o ponto de inflexão da curva..

Para exemplificar vamos mostrar os passos da resolução para a regra:

a(Pessoa):-tosse(Pessoa)<9>, febre(Pessoa)<7>.(1,0).

O processo de resolução para provar a regra "a" consiste, primeiramente em unificar a cláusula "tosse" e obter o valor da sua confiança. O mesmo passo é feito para "febre", e temos as confianças dos resolventes ("tosse" e "febre"). Então o valor da confiança da regra "a" é dado por substituir na fórmula os valores da confiança dos resolventes (as cláusulas "tosse" e "febre"), os valores da adesão(9,7) e os parâmetros de configuração(1,0). Na sequência, o valor da confiança da regra "a" será utilizado no cálculo da confiança da regra "c".

As informações contidas no nível inferior do grafo de conhecimento são fatos que podem ser associados a variáveis linguísticas. Portanto, a implementação destes predicados pode ser feita de modo semelhante aos predicados nebulosos descritos nos modelos anteriores.

Uma dado paciente (a instância da variável "Pessoa" neste caso), possui uma ficha médica com dados que estão colocados na base de conhecimento. Os predicados nebulosos, como por exemplo "febre", funcionam da seguinte forma: é verificado se a "Pessoa" possui alguma informação sobre febre; se o dado (temperatura) existir, obtém-se o valor do seu grau de pertinência no conjunto nebuloso febre. Os elementos e seus respectivos graus de pertinência no conjunto nebuloso são padrões fornecidos pelo especialista e representam a sua concepção do assunto. Assim, como visto nos modelos anteriores, os predicados nebulosos podem ser implementados por funções matemáticas que representam este conceito ou, de acordo com os dados fornecidos, o predicados devem ser capazes de obterem valores intermediários utilizando a interpolação. Portanto, neste modelo todo predicado nebuloso unifica, e a sua confiança é obtida em um conjunto nebuloso. Na ausência da informação sobre febre, assume-se o valor 0.5 ou seja a ignorância do especialista sobre o assunto.

b) Mecanismo de Controle

Os grafos são divididos em três níveis: o nível inferior onde estão os nodos de entrada, o nível intermediário onde estão os nodos associativos e o nível superior onde estão os nodos de decisão.

Ao efetuarmos a conversão dos grafos para as regras na sintaxe PROLOG, passamos a ter somente 2 níveis: as regras de entrada e as regras associativas. O nodo de decisão é representado através de uma regra associativa.

A eliminação de um nível deve-se ao PROLOG representar somente fatos (predicados nebulosos) e regras (regras associativas), e seu mecanismo de controle não ser equivalente as regras de decisão. Um terceiro nível no PROLOG consistiria de um mecanismo de controle com capacidade de tomada de decisão e de aprendizado.

Uma regra associativa sucede somente se o valor da confiança gerada é superior ao valor da aceitação da regra. O limiar de aceitação é facilmente obtido dos especialistas durante a construção do grafo. Portanto, no programa acima a regra "tuberculose", que é uma regra associativa, sucederá se o valor resultante do cálculo da confiança for superior ao limiar de aceitação da regra. As regras de entrada equivalem a predicados nebulosos que sempre unificam e não possuem limiar de aceitação.

O mecanismo de retorno ("backtrack") dentro deste contexto, só tem sentido nas regras associativas, pois os demais predicados (nodos inferiores do grafo) sempre unificam e não possuem limiares de aceitação.

O mecanismo de retorno mantém-se igual ao dos PROLOG-nebulosos. Isto pode ser verificado no programa dado acima ao tentarmos provar o predicado "tuberculose". Ao iniciarmos a consulta, a lista de "backtrak" é montada com as duas regras tuberculose restantes (regra 2 e 3).

Inicialmente iremos provar o corpo da primeira regra "tuberculose" que consiste da cláusula "c". Se o valor obtido no cálculo da confiança de "c" é maior que o limiar de aceitação o predicado "tuberculose" estará provado, pois no corpo há somente a clausula "c", e a confiança da primeira regra "tuberculose" é igual a confiança de "c". Se o valor da confiança de "c" é menor que o limiar de aceitação o predicado "c" falha e o "backtracking" ocorrerá, e a lista de "backtraking" deverá ser consultada para que uma nova cláusula seja tomada. Neste caso o próximo predicado na lista de backtrackng é a segunda regra "tuberculose" dada no programa, que passa a ser a novo predicado a ser provado. Nesta regra a confiança do predicado "tuberculose" é obtida a partir do valor da confiança de "c" e da confiança do predicado nebuloso "bacilus_cultura". Se o valor ultrapassar ao limiar de aceitação a hipótese estará provada. Caso contrário, o sistema deverá buscar (efetuando novamente o "backtrack") a última das regras "tuberculose" . Se em nenhuma das regras "tuberculose" o limiar de aceitação for ultrapassado, a hipótese é rejeitada e a consulta falha.

Portanto este modelo possui valores de limiares em cada regra associativa. Os valores dos limiares são fornecidos pelos especialistas durante o processo de aquisição do conhecimento.

No modelo de PROLOG-nebuloso convencional descrito acima, a prova da regra "tuberculose" nunca poderá ser efetuada se a cláusula "c" não ultrapassar o limiar, pois nenhuma das cláusulas que a contém poderá ser provada devido ao cálculo da confiança ser baseado no valor mínimo. Quando trabalhamos com um cálculo aditívo, mesmo que a confiança da cláusula "c" não atinja um valor maior que o limiar, podemos agregar novos fatos aumentando a sua confiança e ultrapassar o valor do limiar. Portanto, a configuração do operador nebuloso para as duas últimas regras "tuberculose" deve ser feita de forma a ter um comportamento aditivo. A agregação de fatos nas cláusulas demostram que este modelo pode manipular o raciocínio aproximado.

O mecanismo de controle do PROLOG, quando manipula as estruturas como a do programa acima é muito ineficaz. Em todas as regras "tuberculose" existe a cláusula "c". Se na primeira regra a cláusula "c" falhar, na prova da próxima regra ao encontrarmos "c" todos os passos de sua prova serão repetidos. Evidentemente este comportamento não é aconselhável pois causa um baixo desempenho e não corresponde à forma com que os especialistas operam. Podemos ver nos grafos de conhecimento que após

uma informação ("c") ser agregada, durante o processo de prova, não é necessário refazer os passos para obtê-la.

Ao contrário dos PROLOG-nebuloso, que possuem dois operadores (max-min) e necessitam duas representações (normalmente os operadores "," e ";"), o operador aditivo é único (as variações ocorrem devido a sua configuração) e possui uma única representação.

Vimos que o retorno ("backtracking") ocorre somente nas regras que denominamos associativas. Portanto, nestas regras o mecanismo de controle leva em consideração o resultado do cálculo da confiança para determinar qual o próximo passo (próxima regra) a ser executada. Vimos que a lista do ponto de "backtraking" é construida obedecendo a sequência em que as regras são inseridas na base. Evidentemente, a sequência das cláusulas na base é importante para que a ordem da execução seja correta. A ordem das cláusulas nas regras "a" e "b" é indiferente pois nessas regras os fatos sempre unificam e a ordem para o cálculo é indiferente. Nas outras regras a ordem das cláusulas é importante tanto a nível de controle quanto de cálculo.

A sequência das regras pode ser facilmente obtida percorrendo os grafos da esquerda para a direita. É importante frisar que os grafos são construídos pelos especialistas da esquerda para a direita descrevendo a sua sequência de raciocínio para a prova da hipótese. Portanto, ao transportarmos os grafos, na mesma sequência para os programas, estamos obtendo uma representacão consistente e eficiente.

Também no PROLOG-nebuloso convencional é importante a ordem em que as cláusulas são colocadas na regra, pois nela baseiam-se as estruturas de controle.

c) Raciocínio Parcial

Um sistema que manipula raciocínio parcial é aquele que, mesmo sem dispor de todas as informações, é capaz de aceitar uma hipótese ou, quando nenhuma das hipótese alcançou seu limiar de aceitação, é capaz de escolher entre as hipóteses aquela que é a mais promissora de ser pesquisada.

Portanto, o sistema possui dois passos: o primeiro consiste na obtenção dos valores da confiança das hipóteses; e no segundo, baseado na confiança das hipóteses, o sistema irá tomar as hipóteses aceitas (se existirem) ou tentar provar as hipóteses promissoras.

Dado um conjunto de grafos representando uma especialidade (por exemplo as doenças pulmonares na área médica), onde os grafos são como os de tuberculose descrito acima, que representam os passos para a prova de uma hipótese sobre algum assunto específico (tuberculose). Toma-se um conjunto de dados a respeito do assunto (no caso médico, os sintomas), que são propagados nos grafos fornecendo a confiança das hipóteses ativadas (hipóteses que obtiveram algum valor de confiança). Em uma dada hipótese, se todas as informações estiverem presentes e com valores de confiança altos, a hipótese atingirá seu valor de aceitação, e consequentemente a hipótese será aceita. Podemos ter situações onde nem todas as informações estão presentes, porém as informações disponíveis são suficientemente representativas (altos valores de confiança e adesão) o valor de aceitação é alcançado e a hipótese aceita.

Pode-se aceitar mais de uma hipótese. Evidentemente o sistema dará as respostas numa lista com as hipóteses aceitas e sua respectiva confiança.

Em situações em que nenhuma das hipóteses atingiu o limiar de aceitação, toma-se um conjunto contendo as hipóteses rejeitadas que atingiram os maiores valores de confiança durante a fase de propagação. Com este conjunto, o sistema passa a uma segunda fase que pode ser chamada de fase ativa, que consiste em percorrer os grafos das hipóteses selecionadas e questionar a respeito das informações ausentes.

Durante o processo ativo, uma ou mais hipóteses podem atingir o limiar de aceitação e a solução é obtida. Se nenhuma solução é encontrada pode-se dizer que o sistema é ignorante para o conjunto dos dados fornecidos, ou as informações contidas nos grafos não são consistentes e não representam o comportamento desejado, ou a pessoa é sã.

d) Conceituação Prática do Raciocínio Parcial

A parte de propagação das informações na rede é equivalente ao modelo mostrado anteriormente em a) e b).

Vimos acima que para a implementação dos grafos numa sintaxe PROLOG as regras são divididas em dois níveis (associativas e as de entradas). As regras de entradas (equivalentes aos predicados nebulosos) são responsáveis pela manipulação das informações. Quando a informação é ausente o sistema atribui valor 0.5, pois este valor não altera o cálculo. As informações obtidas nas cláusulas de entrada são associadas pelas regras (regras associativas) dos níveis seguintes. Através da propagação das informações nas regras obtemos os valores das confianças das hipóteses e podemos ter alguma das seguintes situações: se o valor de confiança das hipóteses (valor da confiança das regras que possuem o mesmo nome da hipótese, "tuberculose" são

três regras) não foram calculados, significa que a rede foi parcialmente ativada e a hipótese deve ser descartada. Se o valor da confiança da hipótese é maior que o limiar de aceitação a regra sucederá e esta é uma das hipóteses. Se o valor da confiança da hipótese foi calculado e seu valor é menor que o limiar de aceitação (no caso da "tuberculose" toma-se o maior valor da confiança obtida nas três regras hipóteses) então esta hipótese poderá ser reavaliada pelo sistema.

Portanto o nível de propagação pode ser processado em paralelo e deverá fornecer ao segundo nível as hipóteses aceitas e seus valores de confiança, no caso de sucesso na prova das hipóteses; ou fornecer as hipóteses promissoras (aquelas que não ultrapassaram o limiar) e as lista das informações ausentes na prova de cada hipótese; ou a informação que nenhuma das hipóteses teve cálculo de confiança efetuados (isto pode significar que as informações fornecidas estão fora do escopo).

O segundo nível, que chamaremos de nível de decisão, herdará as informações produzidas pelo primeiro nível, e a partir delas executará os passos necessários. Se nenhuma informação é herdada, significa que nenhuma hipótese foi ativada. Se nas informações herdadas existem hipóteses ativadas, o sistema informará se existirem, as hipóteses aceitas na ordem de importância. Se nas informações herdadas não existirem hipóteses aceitas, toma-se a lista das hipóteses promissoras e tenta-se efetuar a prova das hipóteses obtendo-se as informações ausentes.

As informações ausentes são solicitadas através de uma interface com o usuário, pedindo-se uma dada informação, ou em uma base de dados. As informações obtidas são novamente propagadas em todas as hipóteses ou sómente nas selecionadas no segundo nível. Se as informações são propagadas em todas as redes o processo reinicia repetindo-se o nível 1 (um) novamente. Uma outra forma de processamento tentaria apenas provar uma a uma das hipóteses promissoras até obtenção da prova de uma delas. Podemos também ter situações em que nenhuma das hipóteses promissoras é aceita. Neste caso as informações são inconsistentes com o sistema.

Podemos concluir que um sistema que opera com raciocínio parcial deve conter operadores que não refutem uma regra, mesmo que um dos fatos que a compõem estejam ausentes.

Analisando as características dos modelos de PROLOG-nebulosos (convencional e Mukaidono) descritos e as características do modelo aditivo que manipula o raciocínio parcial e o aproximado, podemos concluir que as estruturas de controle desses modelos não são correspondentes. O segundo nível do modelo aditivo não posssui estrutura similar nos PROLOG-nebulosos e a sua criação corresponderia à criação de regras de decisão. Os cálculos adotados nos PROLOG-nebulosos são baseados no max-min que não é aditivo.

Podemos efetuar a análise do grafo nebuloso, mostrado na figura 4.5 de maneira simples através de linguagens orientadas por objetos. Nos grafos vemos que os nodos trocam mensagens (valor do cálculo de confiança) entre si. Os nodos inferiores são objetos que são instanciados, os nodos intermediários e de decisão são objetos que recebem mensagens e possuem cálculos específicos. Baseado nessa análise pode-se afirmar que o paradigma de orientação por objetos é a forma mais eficiente para representar este modelo que o paradigma lógico. Evidentemente, daí tiramos que o paradigma lógico não é a melhor maneira de representar o raciocínio dos especialistas.

CAPÍTULO 5

IMPLEMENTAÇÃO DE UM INTERPRETADOR DO PROLOG NEBULOSO CONVENCIONAL

Descreveremos as principais características e estruturas internas que compõem o PROLOG-nebuloso, enfocando principalmente os aspectos referentes à manipulacão da lógica nebulosa. Este modelo de interpretador é baseado no interpretador PROLOG implementado em Klusniak[KLUS85 e SZPA85], com as alterações necessárias para a manipulação da lógica nebulosa e dos predicados nebulosos.

Nesta implementação não existe preocupação com o desempenho e com mecanismos de recuperação de espaços de memória alocados dinâmicamente durante a execução (conhecido como "garbage colector"). Na solução de problemas complexos o modêlo rapidamente esgotará o espaço de memória disponível. Este modelo está voltado unicamente para estudos dos mecanismos que compõem o PROLOG-nebuloso e não está voltado para aplicações comerciais.

O interpretador está dividido em dois grandes módulos, o núcleo e a interface como visto no desenho abaixo. O Núcleo interpreta uma sintaxe restrita própria. Ao iniciar o funcionamento do interpretador o Nucleo lê, em um arquivo chamado de "boot" os parâmetros para configuração do interpretador, como por exemplo a lista dos predicados do sistema e um predicado "translate" que transforma um programa PROLOG escrito na sintaxe do PROLOG-10 para um programa na sintaxe restrita interpretada pelo núcleo. A interface que é escrita na sintaxe do PROLOG-10 é convertida, pelo predicado "translate", para a sintaxe do núcleo. Em seguida o núcleo carrega o programa interface (já na sintaxe do núcleo) e passa a executar o programa monitor que consiste de um ciclo que primeiramente mostra o sinal de pronto "?-" na espera de alguma consulta, e se ocorrer um consulta à executa . Portanto interface é um programa que interpreta cláusulas na sintaxe do PROLOG-nebuloso (o mesmo do PROLOG 10) transforma para uma lista que obedece a precedência dos operadores e a execução é efetuada pelo predicado do sistema (call) contido no núcleo.

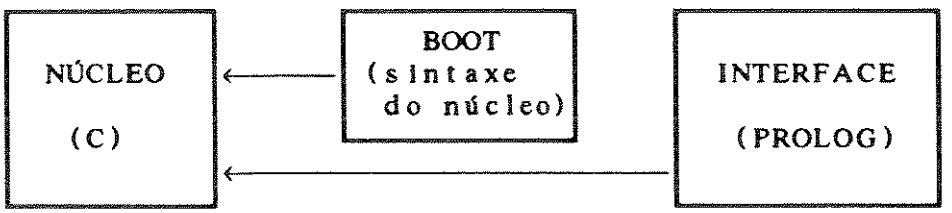


Fig. 5.1 : Principais módulos do Interpretador

5.1 NÚCLEO

O núcleo foi implementado usando a linguagem C, e contém as rotinas responsáveis pela montagem da base de dados, mecanismos de inferência, as rotinas do sistema ("built-in") e os predicados nebulosos.

Efetuaremos a seguir uma descrição bastante ampla das partes que compõem o núcleo a fim de permitir uma visão global. Os detalhamento das principais partes do núcleo será efetuado posteriomente.

As rotinas foram divididas em partes quanto às suas utilidades, e que representam as tarefas efetuadas pelo núcleo. A seguir descrevemos suscintamente as principais partes do núcleo:

- Definições das estruturas de dados utilizadas para: construir a base de dados, armazenar os passos executados pelos mecanismos de controle e armazenar as instâncias geradas durante o processo de unificação. Encontra-se a definição da lista dos identificadores das rotinas do sistema e o código dos erros a serem tratado.

- Todos os erros gerados durante a execução das rotinas do nucleo sofrem um tratamento interno e uma mensagem de alerta é impressa.

- Rotinas que permitem criar e apagar arquivos, e rotinas básicas para ler e escrever em arquivos ou nos canais de entrada e saída. Estas rotinas são as interfaces do PROLOG-nebuloso com o exterior.

- Existem rotinas que efetuam a interpretação das cláusulas na sintaxe restrita e inserem os simbolos e os protótipos nas estruturas de dados especificas. Os simbolos representando os predicados e os parâmetros (por exemplo o valor de pertinência), são inseridas nas estruturas de dados que representam os termos (neste modelo os termos serão representados por partilhamento não estruturado). As operacões efetuadas durante uma consulta, normalmente geram instâncias das cláusulas contidas na base, e as instâncias geradas são inseridas em estruturas de dados apropriadas.

- As rotinas do processo de controle são responsáveis por, dada uma lista de consulta, verificar se a cláusula corrente unifica com as informações contidas na base ou trata-se de rotinas do sistema. Se for uma cláusula e unificar com a base de dados, ela ira gerar as instâncias que são montadas na base de dados. Se for uma rotina do sistema o procedimento referente será executado. Todos os passos do processo de controle são armazenados em estrutura de dados específicas, a fim de permitir a recuperação dos dados após a ocorrência do retorno ("backtrack") ou a implementação de mecanismos extra lógicos. Como as operações efetuadas durante o processo de controle se fazem básicamente sobre as estruturas de dados, foi construído um conjunto de rotinas para acesso e manipulação dos dados nas estruturas (por exemplo os predicados de inserção de informação na base de dados).

- As rotinas do sistema são: procedimentos para as situações nas quais não é possível ou é inconveniente darmos uma tratamento lógico para o problema; e procedimentos para a manipulação dos recursos contidos no núcleo (por exemplo inserir e apagar dados da base). Dentro deste grupo estão os predicados nebulosos que são funções matemáticas às quais podemos associar variáveis linguísticas ou conjuntos nebulosos.

- Possui rotinas que interpretam cláusulas na sintaxe restrita, que consiste na reprodução das cláusulas na forma de uma lista.

A rotina principal do núcleo consiste em um ciclo que lê as cláusulas na sintaxe restrita e monta as mesmas na estrutura de dados. Se é uma cláusula objetivo então busca-se provar a consulta.

5.1.1. Principais Estruturas de Dados

Abaixo descreve-se graficamente as principais estruturas de dados que compõe o Núcleo.

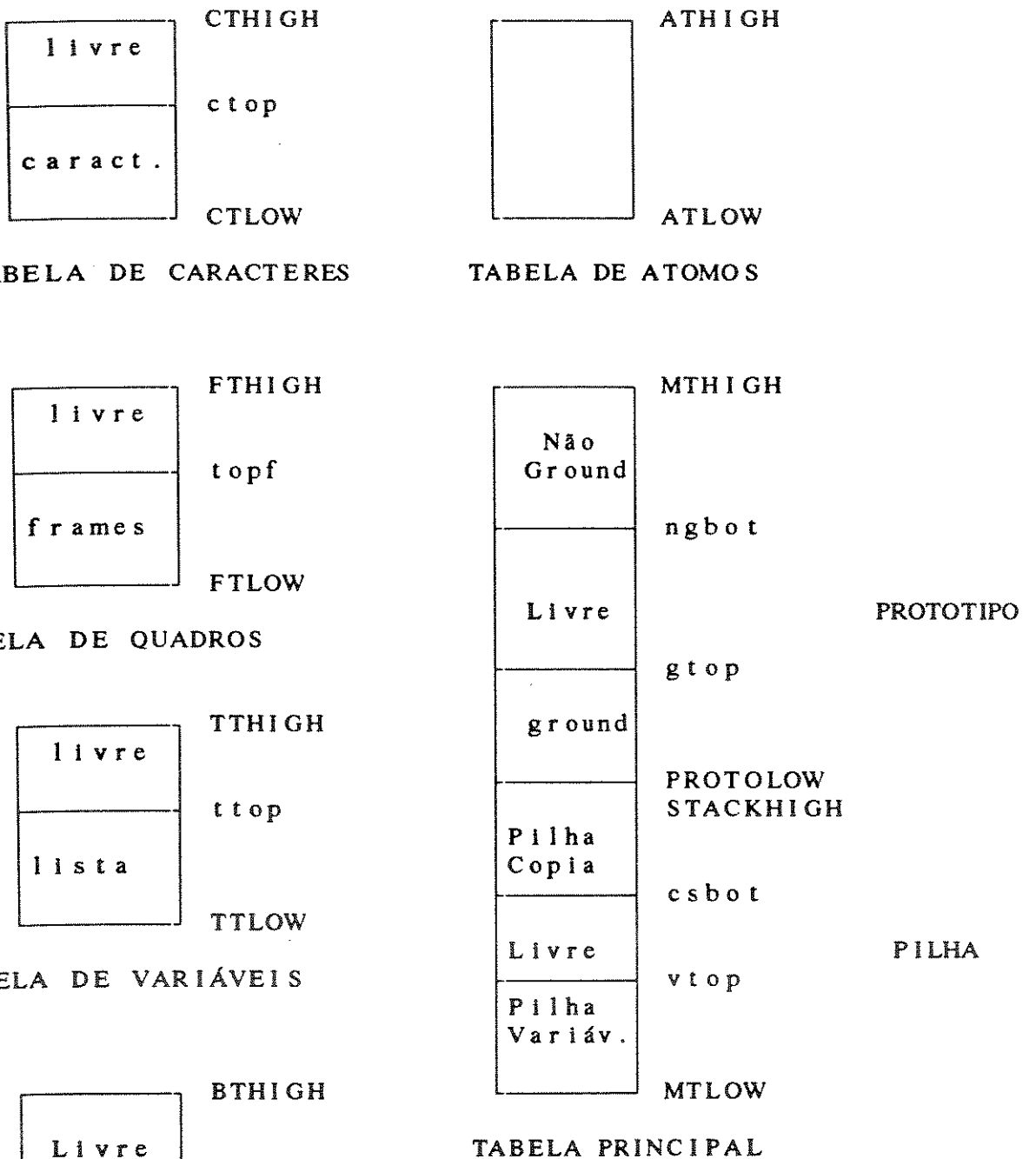


Fig. 5.2. Estruturas de Dados Definidas no Núcleo.

As tabelas são descritas a seguir.

a) Átomos e Descritores

A tabela de caracteres é usada para armazenar todos os nomes das funções e símbolos dos predicados. Esta tabela será chamada de TC.

A tabela de átomos (TA) armazena as estruturas de dados onde são colocadas as informações sobre o simbolo (função ou predicado). A estrutura é composta dos seguintes campos:

- nome do átomo (endereço da TC onde se encontram a cadeia de caracteres),
- número de parâmetros,
- apontador para a estrutura descritora do procedimento.

Estas duas estruturas (TA, TC) são usadas para a tradução da representação de termos e cláusulas da forma externa para interna e vice-versa. Ao intepretarmos um símbolo ele é imediatamente inserido na tabela de caracteres. Podemos, utilizando as estrutura dos átomos, verificar se este símbolo é novo ou não. A busca de símbolos na tabela dos átomos, é efetuada utilizando um algorítmo de "hashing" linear. Se não existe o átomo na tabela, aloca-se espaço em (TA) e monta-se uma estrutura para descrevê-lo, que contém o endereço na tabela TC onde inserimos o nome, o número de parâmetros que ele manipula e o endereço do seu descritor. Se o átomo existe, o nome que foi inicialmente colocado em TC é apagado.

O descritor é a parte onde monta-se as estruturas das cláusulas ou predicados. As estruturas dos descritores são alocadas dinâmicamente e possuem os seguintes campos:

- apontador para a próxima estrutura descritora se existir, ou vazia caso contrário,
- número de variáveis
- número do procedimento no sistema (se é um predicado do sistema), ou um apontador para o protótipo da cláusula cabeça (endereço na tabela principal) e o apontador para a cauda da cláusula (endereço na tabela principal).

Podemos ver a utilização destas estruturas na representação, do predicado "member" nas figuras a seguir. Na figura 5.3 temos os descritores das duas regras e na figura 5.4 temos a representação completa.

O predicado "member" na sintaxe restrita é dado por:

member(:0, :0._):[]

member(:0, _.:1): member(:0, :1).[]

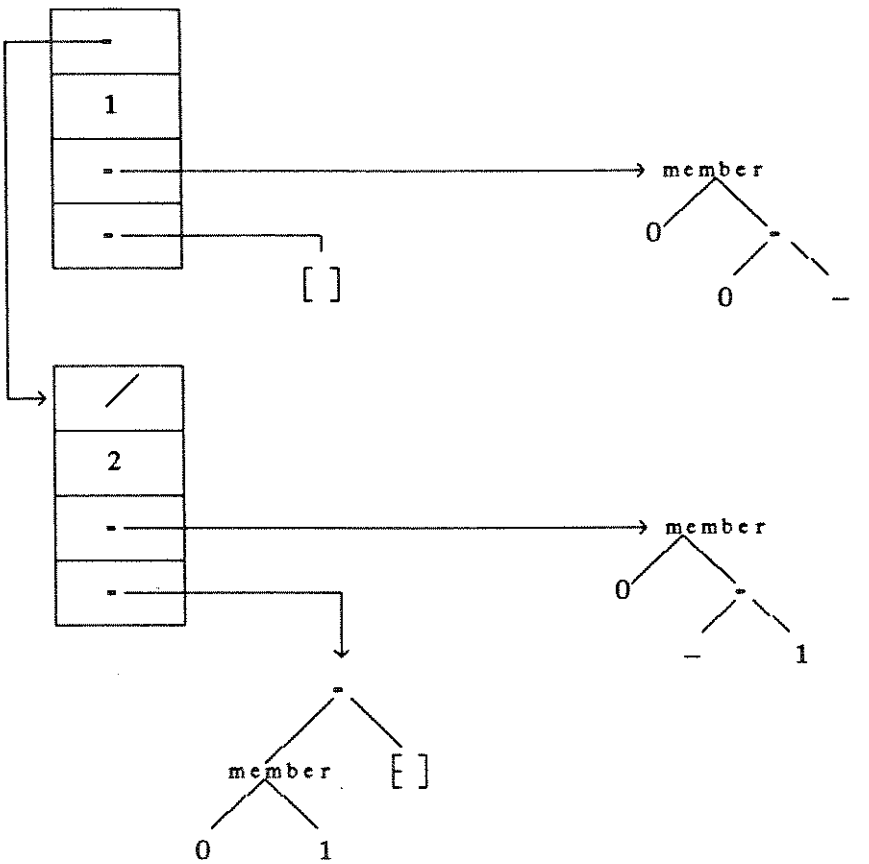


Fig. 5.3: Estrutura do descritor do predicado "member".

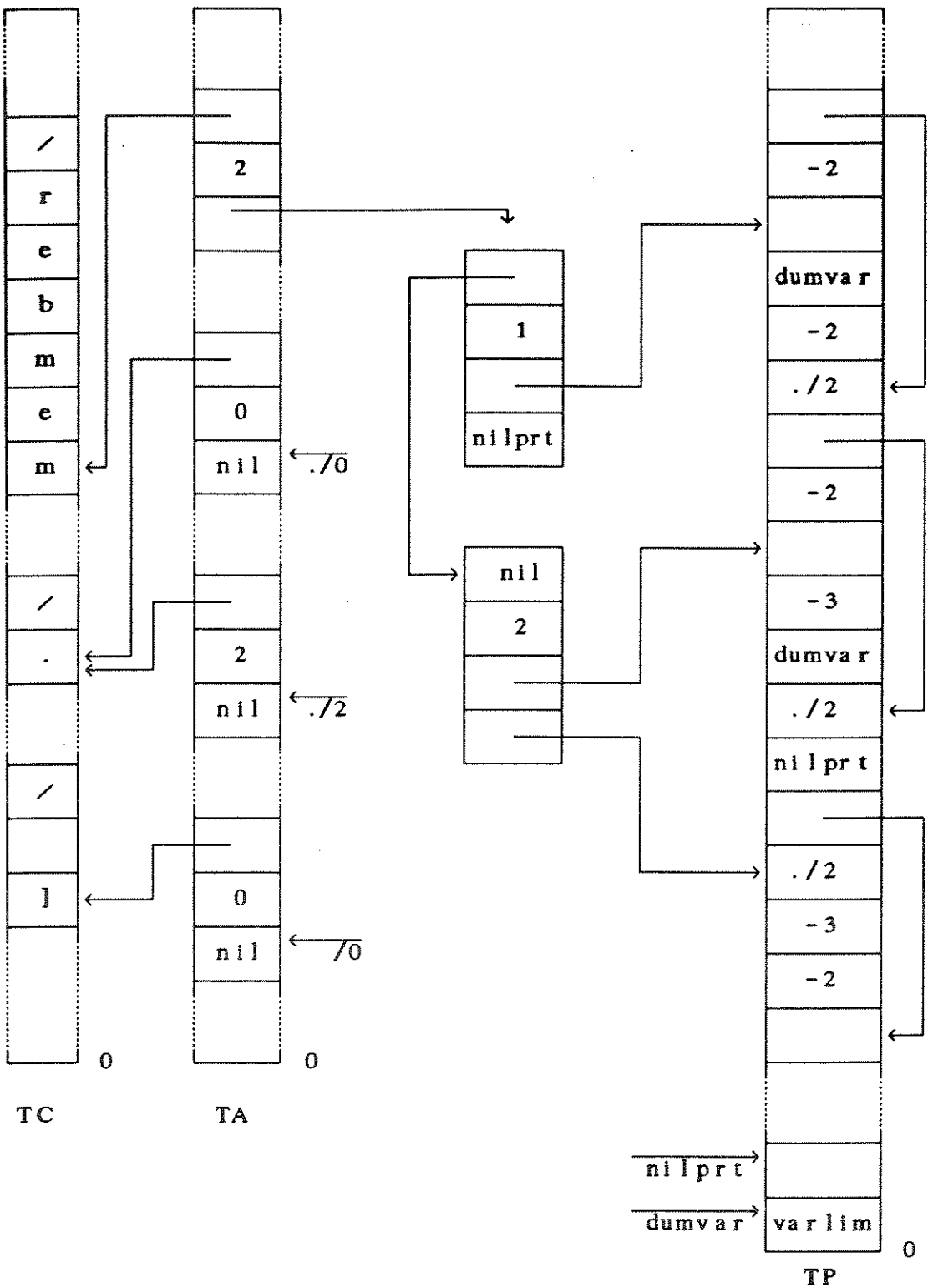


Fig. 5.4 : Estruturas de dados alocadas para a representação do predicado "member".

A tabela principal (TP) é usada para armazenar as instâncias dos termos e os protótipos das cláusulas (veja a Fig.:5.2). Dividem-se em duas áreas: a área de protótipo e a área da pilha. A área do protótipo é dividida em duas partes: local onde armazenam-se os protótipos com variáveis e o local onde armazenam-se os protótipos sem variáveis.

A área da pilha é dividida em pilha de cópia (onde armazenam-se os protótipos instâncias sem variáveis) e a pilha de variáveis (instâncias com variáveis).

Para representarmos os protótipos e as instâncias, necessitamos construir representações distintas para objetos que denotarão inteiros, reais, variáveis e termos normais. Os inteiros são objetos representados por duas palavras onde a primeira contém a marca INT (valor 1) e a segunda o valor correspondente. Os valores reais são utilizados apenas para a representação do grau de pertinência e possuem valor restrito ao intervalo [0,1]. Os valores reais são interpretados e convetirdos para valores inteiros, por esse motivo os valores reais devem ter apenas quatro casas decimais de refino. Os reais são objetos que possuem a marca REAL (valor 2) na primeira palavra e um valor inteiro na segunda palavra, que é convertido para real quando manipulado. Os objetos representando variáveis utilizam somente uma palavra e seu conteúdo são valores negativos. Os termos normais são construídos por uma sequência de palavras, onde a primeira palavra contém o endereço da função na tabela de átomos e as demais são os argumentos. Os argumentos podem conter variáveis ou apontadores para objetos apropriados. As instâncias são representadas por dois apontadores. Se o primeiro apontador for um protótipo, o segundo aponta a pilha de variáveis. Se o primeiro aponta a instância de um termo, o segundo é descartado. A tabela principal é inicializada com protótipo dos inteiros mais comuns (0..10), caracteres e o símbolo de final de lista ([]) a fim de permitir um melhor desempenho e evitar duplicação de protótipos.

Vimos acima as regras do predicado "member" escrito na sintaxe interpretada pelo núcleo. As rotinas do núcleo interpretam estas cláusulas e as informações são inseridas nas tabelas criando uma estrutura de dados tal como vemos na figura 5.4, e nela podemos ver o conteúdo da tabela de caracteres, tabela de átomos, os descritores e a tabela principal. Na figura 5.3, podemos verificar que há dois descritores de procedimentos, um para cada regra de "member" e estão encadeados pois têm o mesmo nome e número de parâmetros diferentes. Nos descritores existem apontadores (posição contendo endereços) para a tabela principal onde estão construídos os protótipos da cláusula cabeça e da cláusula cauda. Na tabela principal (TP) estão montados os protótipos. Nos protótipos os campos ocupados por "member/2" e "./2" representam o

endereço da tabela de átomos. Os valores negativos correspondem as variáveis e as posições que contém apontadores (endereço) para a tabela principal são representadas por setas. As estruturas de dados do PROLOG-nebuloso são as descritas acima e as operações de controle são efetuadas sobre elas. Pode-se facilmente verificar que as estruturas de dados são complexas, de difícil manipulação e consomem muito espaço em memória.

b) Estruturas de Controle

Dado um conjunto cláusulas

(*i*) a(X) :- b(X).

(*iii*) b(Z) :- c(Z), d(Z).

(*iii*) b(f).

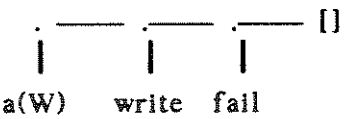
(*iv*) c(e).

(*v*) d(V).

As cláusulas são interpretadas e monta-se nas estruturas de dados, a representação correspondente. Podemos então efetuar uma consulta tal como vemos abaixo, e obtermos as informações contidas na base de dados.

:- a(X), write(X), fail.

Ao efetuarmos a consulta, uma sequência de passos de controle deverão ser executados. Deve-se primeiramente frisar que o corpo das cláusulas é internamente representado como uma lista onde os elementos são protótipos. Portanto, ao efetuarmos a consulta acima o sistema irá gerar internamente a seguinte lista:



O processo de controle consiste em percorrer a lista de consulta tomando-se um a um dos protótipo da consulta e efetuar as unificações com os protótipos contidos na base de dados, como ocorre por exemplo na cláusula "a(W)", ou executar os predicados do sistema, como por exemplo o predicado do sistema "write". Se a lista for totalmente percorrida a consulta sucederá.

A sequência dos passos tomados na prova da lista de consulta e as informações geradas durante a prova são armazenadas nas estruturas de dados de controle.

Na tabela dos quadros implementa-se manipulação da lista de consulta. Os quadros são alocados a cada modificação da lista de consulta e possuem os seguintes campos:

- o quadro anterior,
- apontador para a lista corrente da consulta,
- apontador para pilha de variáveis.

Ao iniciarmos a utilização do sistema a tabela de quadros é inicializada com um quadro contendo uma lista vazia para controle do sistema. Ao efetuarmos uma consulta um novo quadro é alocado (ativação de um quadro) e nele é inserido o endereço da lista de consulta, o quadro anterior e o endereço corrente da pilha de variáveis. Aproveitando a base de dados e a consulta acima, o sistema de controle toma o primeiro protótipo da lista, que neste caso é "a(W)", para provar. Como há na base uma cláusula "a" com mesmo número de parâmetros, a unificação ocorre. Como a cláusula "a" da base possui corpo, um novo quadro é alocado e a lista de consulta neste quadro é o corpo de "a" (a lista passa a ser "b(X)). Este comportamento equivale a substituir na lista de consulta a cláusula "a" pelo corpo da cláusula unificada ("b").

Durante a alocacão do quadro para a consulta, verifica-se que existe uma variável (W) na lista. Aloca-se então na pilha de variáveis uma posição para corresponder a esta variável e seu endereço preencherá o campo do quadro em que colocamos o endereço da pilha de váriáveis. Durante a unificação de "a" as variáveis "X" e "W" passaram a representar o mesmo endereço na pilha de variáveis.

O campo que quarda o quadro anterior é importante. Por exemplo, após a lista "b(X)", gerada pela substituição de "a", unificar com "b(f)", o sistema deverá passar a executar o protótipo "write" e esta informação é obtida pelo sistema no campo que quarda o quadro anterior.

O algoritmo de controle obtém na tabela de quadros as informações de qual é a lista de consulta corrente, o endereço das variáveis contidas na lista naquele momento e o próximo quadro a ser ativado.

A tabela de retorno ("backtrack") é a estrutura onde são colocadas as informações sobre o ponto a partir do qual o algoritmo de controle deverá recomeçar a execução caso ocorra uma falha durante a unificação ou o valor da confiança resultante seja menor que o limiar de aceitação nebulosa.

A tabela é composta de uma estrutura de dados com os seguintes campos:

- endereço do quadro onde o sistema deverá recomeçar,
- valor corrente da confiança (valor verdade do sistema),
- apontador para o descritor de procedimento,
- endereço corrente da pilha de cópia,
- endereço corrente da tabela de variáveis instânciadas.

As estruturas de dados da tabela de retorno são alocadas durante a prova de uma dada cláusula da lista de consulta, quando o sistema depara com mais de uma regra de mesmo nome e número de parâmetros na base de dados. No exemplo acima, ao tentarmos unificar "b(X)" com a base de dados, o sistema verificará que na base existem duas regras com o mesmo nome e que podem ser unificadas. Então aloca-se uma estrutura na tabela de retorno que conterá o índice do quadro corrente na tabela quadros, o valor verdade corrente, o endereço do descritor do protótipo corrente (que neste caso é o endereço do primeiro descritor de "b") e as informações sobre as instânciações correntes das variáveis. Após ter sido montado a estrutura, o sistema passa a prova de "b" unificando com a primeira regra. Se ocorrer uma falha o sistema efetuará o "backtrack", que consiste em descartar as operações efetuadas e recuperar na tabela de retorno as informações anteriores e, neste caso, o sistema irá tomar a segunda regra de "b" para a unificação.

Podemos verificar que não ocorrendo nenhuma falha o sistema deverá obter a primeira solução, pois toma apenas a primeira de todas as regras.

Deve-se frisar que o armazenamento do valor verdade corrente na tabela é uma característica dos PROLOG-nebulosos, não existindo nos PROLOG convencionais.

E finalmente é na tabela de variáveis que são colocadas as variáveis que estão instânciadas. Na verdade, quando uma variável é instânciada tem seu endereço na pilha de variáveis colocado nesta tabela.

5.1.2 Algoritmo de Controle

O algoritmo de controle opera sobre as estruturas descritas acima (estruturas de dados e estruturas de controle). Descreveremos sucintamente a seguir os passos efetuados por este algoritmo.

Passo 0. Inicialmente toma-se a lista de consulta e monta-se na tabela de quadros, o quadro correspondente e inicializa-se com 1 o valor verdade do sistema.

Passo 1. Toma-se o primeiro protótipo da lista de consulta corrente e encontra-se o seu descritor de procedimentos. Se existir mais que um descritor (mais que uma regra na base) é alocada uma estrutura na tabela de retorno onde são colocadas as informações pertinentes (por exemplo o valor verdade corrente).

Passo 2. Se o protótipo é uma chamada do sistema, seu procedimento é selecionado e executado. O proximo passo é o 6.

Passo 3. Se a unificação do protótipo e o descritor sucederem (possuirem mesmo nome, número de parâmetros e confiança maior que o limiar), serão efetuadas as possíveis substituições das variáveis e o valor verdade corrente é comparado com o valor da confiança gerado, passando a ser o menor dos dois valores. O proximo passo é o 5.

Passo 4. Se 2 e 3 falharem efetua-se o "backtrack". Se existir na tabela de retorno alguma estrutura, retiram-se as informações (por exemplo o valor verdade corrente) e reinicia-se em 1. Se não existirem estruturas a prova falha e o algoritmo finaliza indo para 8.

Passo 5. Se existirem protótipos no corpo da cláusula unificada, seu corpo passa a ser a lista corrente. Monta-se, então um novo quadro com a nova lista e as informações correntes e reinicia-se em 1.

Passo 6. Descarta-se o primeiro protótipo da lista (pois já foi executado). Se a lista resultante não é vazia, ela passa a ser a lista corrente e um novo quadro é montado com esta lista e o processo reinicia em 1; se a lista resultante é vazia, toma-se o quadro ancestral. Se o quadro ancestral é o quadro que define o fim da tabela de quadros a consulta sucedeu e finaliza em 7; de outra forma busca-se nos quadros ancestrais: uma nova lista para ser executada montando-se em seguida o quadro e reiniciando o processo em 1; ou o quadro é de fim de consulta e finaliza em 7.

7. Saída após a prova da consulta, com o valor verdade calculado.

8. Saida após a falha da consulta.

5.1.3 Algoritmo de Unificação

Abaixo sera definido o algoritmo de unificação. O algoritmo recebe dois parâmetros (atual e formal), e retorna o valor "verdade" (1) se a unificação ocorrer e "falso" (0), caso contrário.

A unificação sucede quando os parâmetros que descrevem os termos atuais (termos da lista de consulta) e termos formais (descritores do termo) são iguais. Caso contrário, falha.

Na variável "pert" é retornado o valor de pertinência do fato e ou da regra.

```
inteiro unficação(variável termo atual, formal)
variável inteiro sucesso;

 sucesso= verdade;
 se (formal é uma variável) então
     se (atual é uma variável) então
         cria-se uma única representação para as duas variáveis
    senão
          formal é instânciado por atual;
          se (atual é o grau de pertinência)
              então a variável "pert" recebe o valor de atual;
 senão
    se (atual é uma variável) então
          atual é instânciado por formal
          se (atual é o grau de pertinência) então
               a variável "pert" recebe o valor de atual;
          senão
               se (o nome e número de parâmetros da função principal do
                        formal e atual são diferentes) então sucesso = falso
               senão
                        enquanto (SUCESSO e existir argumentos a serem
                                  unificados)
                             SUCESSO = unificação (próximo argumento do
                                atual,  e próximo argumento do formal);
unificação = SUCESSO;
```

5.1.4 Predicados do Sistema

Os predicados do sistema ("built in) são um conjunto de funções que estão à disposição dos usuário. Estas funções são geralmente operações nas quais o paradigma lógico não é eficaz, ou que facilitam a interface entre a máquina e o interpretador.

Os predicados presentes no sistema estão em uma lista cujo o conteudo consiste do nome e número de parâmetros dos predicados. O índice na lista correponderá ao número do predicado no sistema. Ao inicializarmos o sistema esta lista é lida e suas informações são inseridas na base de dados (montagem da tabela de átomos e descritores). Os descritores de predicados neste caso terão o número que correspoderá à função no sistema. Este número será utilizado durante o processo de controle para selecionar a funcão que deve ser executada.

As funções são construídas na linguagem C, e a passagem de parâmetros entre as funcões e o ambiente do interpretador e vice versa são efetuadas por uma estrutura de dados específica.

Podemos introduzir novos predicados no sistema. Isto pode ser feito acrescentando na lista dos predicados seu nome e o número de parâmetros, e o código fonte da função que implementa o predicado. Em seguida o Núcleo deve ser recompilado.

Um dos predicados existentes, que é característico do PROLOG-nebuloso, é o predicado "nebuloso". Ele possui 3 parâmetros

- número da função matemática a qual ele irá se associar;
- um valor contido no universo do discurso (geralmente um inteiro); e,
- valor real referente ao grau de pertinência.

O usuário poderá escolher entre as funcões matemáticas implementadas a que melhor se adapta ao seu problema. Podemos obter, dado um valor do universo do discurso, o seu grau de pertinência na função. O valor obtido é retornado instânciando a variável contida no terceiro parâmetro do procedimento.

5.2 INTERFACE

Vimos na descrição do núcleo, que este é um processador de listas, onde a sintaxe é bastante restrita e o número de predicados disponíveis é pequeno (restringe-se aos predicados do sistema).

Para incrementar o sistema foi criado uma interface que possibilita ao usuário construir seus programas na sintaxe do PROLOG-10 e ter à disposição uma biblioteca de predicados e operadores, e permitindo mecanismos amigáveis para efetuar consulta à base de dados e obter as respostas.

A descrição da sintaxe interpretada pela interface (sintaxe do PROLOG-10) e as principais características da interface podem ser vistas na seção 4.1.2 (A linguagem PROLOG). Portanto, nos restringiremos nesta seção a fazer alguns comentários a respeito da implementação da interface.

A interface consiste de um programa PROLOG, escrito na sintaxe do PROLOG-10. Evidentemente o programa interface utiliza os predicados do sistema contidos no Núcleo e além de outros construídos nele próprio.

Para implantar a interface foi necessário construir um predicado, escrito na sintaxe, chamado de "translate" que é responsável pela tradução de um programa na sintaxe do PROLOG-10 para a sintaxe restrita. Então, utilizando este predicado, a interface foi traduzida para a sintaxe do Núcleo. Após a tradução a Interface foi interpretada pelo Núcleo e colocada em funcionamento utilizando o Núcleo.

A parte da Interface responsável pela comunicação com o usuário é chamada de monitor. Consiste de um ciclo onde o primeiro passo é a colocação do "prompt" "?-" na espera de um comando ser digitado pelo usuário. Os próximos passos são a leitura e execução do comando e a impressão das informações resultantes.

O comandos quando prefixados por ":-" denotam um comando; sem prefixos, denotam uma consulta.

O passo de leitura consiste em converter a cláusula digitada para uma lista, onde a ordem das cláusulas e dos operadores depende da precedência e das prioridades dos operadores, e uma outra lista, contendo os nomes das variáveis contidas na consulta.

Supondo que seja feita a seguinte consulta:

```
?-pai(X,Y),homem(X);homem(Y).
```

Ordenando de acordo com a precedência dos operadores obemos a seguinte lista

```
(';'(','(pai(:0,:1),homem(:0)),homem(:1)),
```

e a execução é efetuada pela chamada do predicado do sistema "call", utilizando os operadores binário ',' e ';' que são construídos para funcionar da seguinte forma

```
','(:0, :1) : call(:0) . call(:1) . []
';'(:0, :1) : call(:0) . []
';'(:0, :1) : call(:1) . []
```

A execução inicialmente consiste em efetuar-se a unificação com o operador ';' e em seguida efetuar-se a execução do primeiro parâmetro (',(pai(:0,:1),homem(:0)), que consiste da execução do operador ','.

Podemos verificar que a implementação do operador ';' (ou) não funciona com o máximo valor das confianças, pois basta que suceda a chamada do primeiro parâmetro para o operador ';' progredir. Para implementar corretamente o operador máximo deve-se calcular a confiança dos dois parâmetros e obter o maior deles.

Após a execução dos comandos ou da consulta o sistema imprime o valor verdade resultante e os valores das variáveis instânciadas, se as mesmas existem.

Existe na interface predicados para a conversão dos predicados em PROLOG 10 para a sintaxe restrita (translate), e a inserção destes predicados na base de dados (consult, reconsult...), e a montagem das cláusulas no formato da lista acima para execução.

A biblioteca é composta por um conjunto de operadores (definição da prioridade e a precedência). Novos operadores podem ser definidos, bastando que sejam inseridos seus predicados e as definição da prioridade e da precedência do operador.

Além dos operadores há um conjunto de predicados para manipulação de cláusulas na base de dados (assert, retract,...), predicados para a listagem dos predicados da base de dados (listing) e predicados para impressão de dados formatados (write...).

A descrição completa dos predicados e da Interface podem ser encontrada em [KLUZ85 e SZPA85].

5.3 EXEMPLOS DE UTILIZAÇÃO

A seguir descreveremos os passos para a montagem de um programa no PROLOG-nebuloso e a utilizacão do interpretador.

Inicialmente, em um arquivo com extensão "pro", criamos através de um editor de texto as cláusulas que compõem o programa.

Como exemplo, escreve-se o programa novatos.pro, e as cláusulas são as descritas a seguir.

```
novato(Pessoa):-idade(Pessoa,Y), novo(Y,K).
idade(maria,30).
idade(glauco,28).
idade(adriana,23).
novo(30,0.6).
novo(28,0.7).
novo(24,0.9).
novo(23,0.95).
novo(20,1.00).
end.
```

O programa é finalizado com o termo "end" e nota-se que aos fatos temos associado seu grau de confiança.

O arquivo contendo o programa PROLOG-nebuloso e o arquivo "proneb.exe", contendo o executável do interpretador, devem estar no mesmo diretório (em sistema operacionais que utilizam esses conceitos).

Ao iniciarmos a execução do interpretador, será mostrada uma mensagem dizendo que o programa está carregando a interface e em seguida é mostrado um "prompt" como abaixo.

```
?-
```

Então, podemos carregar o programa para a base de dados do interpretador utilizando o predicado "consult".

```
?-:-consult(novatos).
```

O sistema mostrará no vídeo os predicados lidos e novamente o "prompt" será mostrado. Então podemos efetuar consulta à base de dados, como por exemplo:

```
?-:-novato(maria).
```

Obtemos a resposta abaixo.

```
valor nebuloso 0.6
sim
?-
```

Se desejarmos saber todas as possíveis respostas para uma dada consulta, efetuamos uma consulta sem o operador ":-" precedendo o comando e a cada resposta tecla-se ";" para se obter a proxima solução.

```
?-novato(X).
X='maria'
valor nebuloso 0.6 ;
X='glauco'
valor nebuloso 0.7 ;
X='adriana'
valor nebuloso 0.95 ;

valor nebuloso 0.00
nao
?-
```

Para finalizamos a execução do interpretador executamos o predicado "halt".

```
?-halt(fim).
```

CAPÍTULO 6

CONCLUSÃO

A proposta inicial do trabalho consistia na construção de ferramentas de programação, que facilitassem ao programador construir sistemas inteligentes utilizando a teoria dos conjuntos nebulosos dentro do contexto da Inteligência Artificial.

No trabalho efetuou-se a análise de todos os modelos de ferramentas de programação baseados na teoria dos conjuntos nebulosos existentes, e propôs-se através do modelo Aditivo, novas idéias e novos caminhos para este tipo de ferramenta.

As primeiras ferramentas analisadas foram as ferramentas de manipulação de conjuntos e relacões nebulosas. Essas ferramentas consistem de um conjunto de funções, que são implementações das operações mais importantes da teoria dos conjuntos e relações nebulosas. Essas ferramentas são úteis para implementar sistemas inteligentes que operam sobre conjuntos e relações nebulosas discretas.

Este sistema não possui um mecanismo de inferência eficiente. Podemos como vimos, criar funções que implementem mecanismos de inferência (teoria da possibilidade). Esses mecanismos de inferência são construídos utilizando as funções contidas no sistema(projeção, composição, etc).

A principal inconveniência desse sistema é que todas as operações são efetuadas sobre conjuntos e relações. Consequentemente, em situações onde manipulamos apenas elementos dos conjuntos ou relações é efetuado um grande número de operações desnecessárias.

Podemos maquiar essa situação criando uma interface onde as operações são efetuadas sobre elementos de conjuntos ou relações. Entretanto, o processamento continuará sendo efetuado sobre conjuntos e relações.

Podemos concluir que estes sistemas são ineficientes quando desejamos manipular uma única informação de um conjunto ou relações nebulosa. Podemos também afirmar que possuem mecanismos de inferência restritos.

O segundo modo de construir ferramentas de programação nebulosa consistiu na utilização do paradigma de programação em lógica.

O primeiro passo consistiu na utilização da linguagem PROLOG para construir programas que manipulassem informações e a lógica nebulosa. Alguns inconvenientes foram notados:

- os programas são deselegantes e difíceis de compreender, pois a manipulação de dados imprecisos não é efetuada de maneira natural e as operações de cálculo não estão associados à linguagem, tornando-se necessária a criação de predicados para manipulação da lógica nebulosa.

- os mecanismos de controle do PROLOG não são efetuados sobre os dados nebulosos, pois estes não são partes implícitas da linguagem, gerando soluções incoerentes.

- ao adicionarmos lógica nebulosa a um dialeto PROLOG certamente perde-se o desempenho, pois inserimos uma nova camada entre o PROLOG e a aplicação.

As soluções para esses inconvenientes foram a criação de uma ferramenta de programação com a mesma sintaxe e semântica do PROLOG, diferenciando-se por manipular a lógica nebulosa e informações imprecisas.

Para a implementação dos modelos de PROLOG-nebulosos foi necessario definir um princípio de resolução nebulosa.

O modelo que denominamos de PROLOG-nebuloso convencional, possui um princípio de resolução que opera apenas com valores no intervalo (0.5, 1.0] (valores positivos). Por manipular somente valores em (0.5, 1.0] as soluções são valores também neste intervalo.

Já no modelo Mukaidono o princípio de resolução opera sobre valores em [0,1], isto devido à utilização de um novo cálculo de confiança baseado no máximo e mínimo.

Neste modelo vimos uma das principais características do PROLOG-nebuloso, que consiste em, todo fato sempre unificar com a base. Os dados para a unifição são obtidos através de interpolação ou através da associação do predicado a uma função de pertinência.

Os PROLOG-nebulosos são úteis na construção de sistemas inteligentes baseados em lógica nebulosa. O PROLOG-nebuloso permite-nos representar dados imprecisos na base de dados, e através do mecanismo de resolução podemos inferir informações da base. O mecanismo de busca dos PROLOG-nebulosos permite-nos obter a primeira solução contida na base. Com a utilização do limiar obtemos soluções maiores que o mesmo. Portanto, temos o comportamento dos conjuntos de nível-α, onde α equivale ao valor do limiar. Evidente que o PROLOG-nebuloso possui mecanismos para obter todas as soluções possíveis.

Como o PROLOG-nebuloso permite o reaproveitamento de todos os programas desenvolvidos para o PROLOG convencional e possibilita a manipulação de informações parciais. Pode-se afirmar que o PROLOG-nebuloso é uma ferramenta mais completa e útil que o PROLOG convencional.

Existem algumas aplicações em que os mecanismos do PROLOG-nebuloso mostram-se inconveniente para sua solução. Por exemplo, não podemos, utilizando o PROLOG-nebuloso construir modelos que manipulam raciocínio parcial e aproximado.

As limitações dos PROLOG-nebuloso ocorrem devido aos operadores nebulosos utilizados (max-min) e o mecanismo de controle do PROLOG.

Utilizando um operador nebuloso aditivo, que possibilita entre outras a modelagem do raciocinio aproximado e parcial, efetuou-se a definição de uma ferramenta de programação baseada nos mecanismo do PROLOG.

Primeiro ponto a destacar é que foi possivel definir um princípio de resolução nebuloso com o operador aditivo. Ao definirmos um PROLOG-nebuloso com cálculo aditivo ele torna-se capaz de efetuar o raciocínio aproximado (à medida em que os dados vão sendo agregados na sequência da prova o valor da confiança vai aumentando). Portanto, para obtermos o raciocínio aproximado podemos utilizar o mecanismo do PROLOG-nebuloso e um cálculo que seja aditivo na medida que os dados vão sendo agregados.

Vimos ser impossível obter o raciocínio parcial (nem todos os predicados devem estar presentes na regras para efetuar o cálculo da confiança) em uma estrutura do tipo PROLOG. A solução é a definição de um novo nível de processamento contendo as regras de decisão.

Percebe-se que à medida em que aumentamos a capacidade das ferramentas baseadas em lógica nebulosa e concebida dentro da estrutura PROLOG, as mesmas tendem a acentuar as limitações da estrutura PROLOG. Com esses dados pode-se afirmar que o mecanismo do PROLOG não corresponde com a forma em que os dados são processados pelo raciocínio das pessoas.

Tendo o paradigma lógico sido esgotado, análises superficiais tem mostrado que o paradigma de orientação por objeto parece ser um caminho promissor e que deve ser explorado na construção de ferramentas de representação e inferência com a teoria dos conjuntos nebulosos.

Com as implementações dos modelos buscou-se ter uma noção das dificuldades e limitações de cada modelo. O modelo de manipulação de conjuntos e relação é um pouco restrito devido ao desconhecimento (não possuia um visão ampla da área) de alguns conceitos da teoria dos conjuntos nebulosos bem como pela dificuldade em criar uma estrutura de dados que não fosse complexa e sim eficiente na representação de informações parciais.

A implementação do modelo PROLOG-nebuloso (modelo convencional) é complexa, devido tanto aos mecanismos de manipulação das estruturas de dados, bem como ao fato dos mecanismo de controle serem de dificíl construção.

Pudemos perceber que uma simples alteração do cálculo pode causar um conjunto de modificações na estrutura do PROLOG-nebuloso. Isto porque geralmente associado ao cálculo existem um conjunto de parâmetros ou uma forma de operar sobre os dados. Isto significa que não podemos ter, ao mesmo tempo, uma estrutura PROLOG e um conjunto de operadores à disposição dos usuários.

Depois de analisar todos os modelos, acredito que os mesmos estão distantes de uma ferramenta de programação de uso geral, a qual possamos utilizar para modelar o processamento das informações durante o raciocínio humano.

As pesquisas de Redes Neurais surgem como um novo possível caminho para as pesquisas desta área. O paradigma de orientação por objetos surge como um paradigma a ser pesquisado.

BIBLIOGRAFIA:

[ADAM80a] Adamo, J. M., "L.P.L a fuzzy programming language I. Syntatic aspects", Fuzzy Sets and Systems, 3 (1980), 151-179.

[ADAM80b] Adamo, J. M., "L.P.L a fuzzy programming language II. Semantics aspects", Fuzzy Sets and Systems, 3 (1980), 261-289.

[ADAM83] Adamo, J. M., " Some applications of L.P.L language to combinatorial programming", Fuzzy Sets and Systems, 6(1983), 43-59.

[BALD87] Baldwin, J. F., "Evidential Support Logic Programming", Fuzzy Sets and Systems, 24(1987), 1-26.

[BALD86] Baldwin, J. F., "Automated Fuzzy and Probabilistic Inference", Fuzzy Sets and Systems, 18, 219-235(1986).

[BALD84 e ZHOW84] Baldwin, J. F., Zhow, S. Q.," FRIL : A fuzzy relational inference language", Fuzzy Sets and Systems, 14(1984), 155-174.

[CAMP85] Campbell, J. A., "Implementation of Prolog", Ellis Hornned Limited, 1985.

[CASA87 et alli] Casanova, M. A., Giorno, F. A. C. e Furtado, A. L, "Programação em lógica e a linguagem Prolog", Edgard Blucher Ltda, 1987.

[CHAN75] Chang, C. L., "Interpretation and execution of fuzzy programs", in : L. A. Zadeh, K. S. Fu, K. Tanaka and M. Shimura, eds., Fuzzy Sets and Their Applications to Cognitive and Decision Processes, 191-218.

[CHAN72] Chang, S. K.,"On the execution of fuzzy programs using finite state machine", IEEE Trans. Computers, 21(1972), 241-253.

[CLAR82 e TARL82] Clark, K. L. and Tarnlund, A. S., "Logic Programming", Academic Press, 1982

[CLOC84 e MELL84] Clocksin, W. F. and Mellish, C. S., "Programming in Prolog", Springer Verlag, 1984.

[DUBO87 e PRAD87] Dubois, D. and Prade, H., "Necessity measures and the resolution principle" , IEEE Transactions on Systems, Man and Cibernetics, vol 17, 13, 474-478(1987).

[FARR82 e PRAD82] Farreny, H., Prade, H., "Make man-machine communication easier: fuzzy programming", Nouvel Automatisme, 27 (1982), 62-67.

[HIND86] Hinde, C. J., "Fuzzy Prolog", International Journal Man-Machine Studies, 24 (1986), 569-595.

[HOGG84] Hogger, C. J., " Introduction to Logic Programming", Academic Press, 1984.

[KAUF75] Kaufman, A., "Introduction to the theory of fuzzy subsets", Academic Press, New York, 1975.

[KLUZ85 e SZPA85] Kluzniak, F., Szpakowicz, S., "Prolog for Programmers", Academic Press, New York,1985.

[LEAO90 e ROCH90] Leão, B. F., Rocha, A. F. "Proposed Methodology for Knowledge Acquisition: A Study on Congenital Heart Disease Diagnosis", Methods of Information in Medicine, 29 (1990), 30-40.

[LEE72] Lee, R. C. T., "Fuzzy Logic and the Resolution Principle", J. Assoc. Comput. Mech., 19(1972), 109-119.

[MART87 et alli] Martin, T. P., Baldwin, J. F. and Pilsworth, B. W. "The implementation of FPROLOG - A Fuzzy Prolog interpreter", Fuzzy Set and Systems, 23(1987), 109-119.

[MUKA87 et alli] Mukaidono, M., Shen, Z. and Ding, L., "Fuzzy Prolog", 3th International Fuzzy Systems Symposium, 5(1987).

[NEGO85] Negoita, C. V., "Expert Systems and Fuzzy Systems", Benjamin / Cummings P. C., 1985.

[NEGO75 e RALE75] Negoita, C. V. and Ralescu, D. A., " Applications of Fuzzy Sets to Systems Analisys", John Wiley & Sons, New York, 1975.

[OGUN81] Oguntade, O. O., "Pragmatics aspect of fuzzy programming", International Journal of Systems Sciences, 12(1981), 135-149.

[OSTA82] Ostasiewicz, W., " A new approach to fuzzy programming", Fuzzy Sets and Systems, 7 (1982), 139-152.

[RAMA87 e SHEK87] Ramamoorthy, C. V., Shekhar, S., Garg, V., "Software development support for A. I. programs", IEEE Computer, January, 1987.

[RICH88] Richards, B. L., "When Facts Get Fuzzy", Byte Magazine, April 1988, 285-290.

[ROCH89 et alli] Rocha, A. F., Theoto, M., Guilherme, I. R., Laginha, M. P., Machado, R. J., "Handling uncertainty in medical reasoning", 5 IFSA, Seatle, USA, 1989.

[RUBI84 e NARA84] Rubin, P. A., Narasinham, R., "Fuzzy goal programming with nested priorities", Fuzzy Sets and Systems, 14, 115-129.

[SANT76] Santos, E. S.,"Fuzzy and probabilistic programs", Information Sciences, 10 (1976), 331-345.

[STEF87 e SAGE87] Stefhanou, H. E. and Sage, A. P., " Perspective on Imperfect information processing", IEEE Transactions on Systems, Man and Cibernetics, vol 17, 5, 1987, 780-798.

[WARR80] Warren, D. H. D., "Logic Programming and Compiler Writing", Software Pratice and Experience, vol 10, 97-125(1980).

[UMAN78] Umano, M., Mizumoto, M., "FSTDS System: A Fuzzy-set Manipulation System", Information Sciences, 14(1978),115-159.

[ZADE65] Zadeh, L. A., "Fuzzy Sets", Information and Control, 8(1965), 338- 353.

[ZADE88] Zadeh, L. A., "Fuzzy Logic", IEEE Computer, Outubro, 83-93(1988).

[ZADE68] Zadeh, L. A. "Fuzzy Algorithms", Information and Control, vol 12, 94-102(1968).

[ZEMA85 e KAND85] Zemankova, M. and Kandel, A., "Implementing Imprecision in information systems", Information Sciences, vol 37, 107-141(1985).