**UNIVERSIDADE ESTADUAL DE CAMPINAS**

**FACULDADE DE ENGENHARIA ELÉTRICA**

**DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL**

# Núcleo de Um Sistema de Processamento de Conhecimento em Tempo Real

Ricardo Ribeiro Gudwin

Prof. Dr. Fernando Antonio Campos Gomide

Tese apresentada ao Departamento de Engenharia de Computação e Automação Industrial da Faculdade de Engenharia Elétrica da UNICAMP, como parte integrante dos requisitos para obtenção do grau de Mestre em Engenharia Elétrica.

**Campinas, 1992**

# Agradecimentos

Agradeço a todos aqueles que me incentivaram e apoiaram na elaboração deste trabalho, especialmente minha família: minha mãe,minha avó, meus irmãos, minha esposa Helena, meus tios, primos e sogros,meus colegas de laboratório, que presenciaram todo o desenvolvimento aqui realizado, ao prof. Márcio, por sua colaboração essencial, e ao meu orientador, que sempre esteve presente nos momentos cruciais, dando-me empenho para a conclusão deste trabalho.

Dedico esta obra especialmente àqueles que não puderam ver este trabalho concluído, mas que sempre me incentivaram na disciplina e na perseverança, ensinando-me que é por meio do trabalho honesto e responsável, pelo empenho e dedicação, que se atingem as grandes metas: A meu pai e meus avôs.

# Resumo

Neste trabalho, desenvolve-se a especificação, implementação e validação de um sistema de processamento de conhecimento com vistas a aplicações em tempo real.

Os sistemas de processamento de conhecimento vêm sendo largamente utilizados em sistemas de automação e controle, principalmente em casos onde o modelo que se tem para o processo ou é muito complicado, o que o torna intratável, não é conhecido, ou o mesmo tem seus parâmetros modificados com o passar do tempo. Apesar disso, pouco se tem feito para adequar os modelos existentes de processamento de conhecimento, aos requisitos de tempo real.

Este trabalho tem por motivação dar uma contribuiçãao neste sentido, criando um procedimento de inferência mais eficiente, que não considere alguns aspectos de menor importância, não necessários em aplicações de tempo real, e que acabam por pertubar o desempenho de tais sistemas quando o tempo real se faz uma necessidade.

Desenvolveu-se um procedimento de inferência matricial, onde um conjunto de regras é transformado em duas matrizes distintas. Estas matrizes são então processadas por meio de operadores especiais, de modo a efetuar a inferência. Este procedimento permite a utilização de regras do tipo proposicional, de predicados de 1ª ordem, regras nebulosas e fator de certeza. Permite também uma utilização híbrida entre termos de diferentes tipos.

Além disso, desenvolveu-se dois procedimentos para manipulação de bases de regras do tipo matricial: o procedimento de expansão horizontal e o procedimento de compressão de uma base de regras, colocados de modo a aumentar o desempenho do procedimento. O procedimento é então discutido diante de algumas possíveis implementações computacionais, e é apresentado um exemplo da aplicação do procedimento, para o caso do controle supervisório de grupos de elevadores.

# Índice

# 1. INTRODUÇÃO

## 1.1 Prólogo

Desde o aparecimento do computador como uma ferramenta básica de trabalho, desejou-se de alguma forma trazer para a máquina um pouco daquilo que os seres humanos chamam de inteligência.

Esta tarefa revelou-se, entretanto, de especial dificuldade visto que o conhecimento humano não é de fácil modelagem e compreensão. Aquilo que parecia ser uma grande promessa no campo das ciências foi relegado a um segundo plano, diante da eficiência com que os computadores realizavam cálculos matemáticos, o que parecia ser uma vocação natural destas máquinas.

Com isso, mais e mais todo o ferramental matemático utilizado na elaboração de circuitos eletrônicos sofisticados foi sendo transferido para os sistemas computadorizados, sendo utilizados em vários sistemas, como os de controle de processos e automação da manufatura.

Entretanto, a medida que os sistemas tornavam-se mais sofisticados, ía-se delimitando sua ineficiência em tratar alguns tipos de processos. Tais processos fugiam muito do habitual assimilável pela teoria de sistemas corrente, e tornavam uma descrição não linear tão complicada que não conseguia-se modelar matematicamente os mesmos, sem incorrer em simplificações que deturpavam a descrição real do processo. Isso tornava praticamente impossível a implementação de modelos adequados para tais processos a nível de planejamento e controle.

Foi neste contexto que percebeu-se que apesar dos processos terem um modelo matemático incrivelmente sofisticado, os mesmos eram habilmente manipulados pelo ser humano, sendo frequentemente descritos em termos de linguagem natural.

Dentro deste contexto, a estruturação do conhecimento humano e sua representação em uma linguagem matemática tornou-se inequivocamente necessário, de modo que o mesmo pudesse ser manipulado e com isso fosse possível conseguir os resultados que eram impossíveis (ou de implementação excessivamente sofisticada) para a teoria matemática convencional de modelagem e controle de processos.

Esta motivação deu um grande impulso para as pesquisas na área, o que foi sendo corroborado com a aparição dos primeiros resultados práticos a nível de controle e automação.

Hoje encontra-se consolidada a importância dos sistemas de processamento de conhecimento na área de automação e controle [14,17,18,30,31,34], principalmente diante de sistemas de caráter intrinsecamente não linear e onde a teoria de sistemas lineares ou mesmo o

pouco que existe para sistemas não lineares não é facilmente aplicável. Nestes, a aplicação dos ditos "sistemas inteligentes" tem gerado resultados favoráveis, fomentando o investimento em pesquisas no tema, com resultados promissores [16,20,24,40].

Por ser uma área de estudos relativamente recente, a pesquisa em inteligência artificial tem ainda uma grande diversidade de pontos que não estão totalmente esclarecidos. Assim, denota-se a existência de uma vasta área de trabalho no âmbito de pesquisa acadêmica ou mesmo pesquisa aplicada. Isso vem sendo explorado de modo a contribuir para a consolidação da inteligência artificial como um método científico bem estruturado e que possa ser aceito uniformemente pela comunidade científica.

A medida que as pesquisas evoluem (e ecos destas evoluções fazem-se ouvir de todos os cantos) percebe-se o quanto ainda se está distante de se atingir o propósito inicial de mimetizar a inteligência humana por meio de máquinas [3]. Entretanto, cada vez mais percebe-se que o pouco que já existe em termos de processamento do conhecimento mostra-se absolutamente útil e aplicável na solução dos mais diversos tipos de problemas.

## 1.2 Representação do Conhecimento

A idéia básica por trás do uso de sistemas de processamento de conhecimento é a de que pode-se estruturar o conhecimento humano, ou pelo menos uma certa parte deste conhecimento, em alguma linguagem formal. Estando um conhecimento útil representado de algum modo, em uma linguagem formal, espera-se através de uma certa manipulação sobre ele, obter-se conclusões ou resultados que possam ser então utilizados nos sistemas de controle ou de tomada de decisão em geral [38].

Uma das primeiras formas que utilizou-se para representar o conhecimento humano e traduzí-lo para uma linguagem formal foi através da lógica matemática [7,9]. Diversos tipos de lógica foram então propostos, dentre eles a lógica proposicional, a lógica de predicados de primeira ordem [29], a lógica temporal [35] e mais recentemente a lógica nebulosa (*fuzzy logic*) [8,25,41]. Outros métodos foram propostos, tais como o uso de redes semânticas, frames, linguagens orientadas a objetos [33] e mais recentemente, abordagens coneccionistas [10,13] tais como as redes neurais e os algoritmos genéticos [6]. Tais métodos as vezes pecam por carecer de um maior formalismo ou rigor matemático, ou então têm uma aplicação muito restrita (como é o caso das redes semânticas), apresentando-se entretanto como alternativas interessantes e dignas de maior estudo por parte de pesquisadores da área. Particularmente a técnica de *frames* vem sendo bem explorada e apresenta alguns resultados interessantes.

Neste trabalho, será adotado um caráter mais voltado para a lógica, visto que ela é definida como uma linguagem formal bem sistematizada, o que permite desenvolvimentos com uma base formal sólida.

Utilizar-se-á de um esquema proveniente da inteligência artificial conhecido como "Sistemas de Produção" [28], onde privilegia-se de certa forma a lógica proposicional e a lógica nebulosa. O uso de regras de produção proposicionais será encarado como um objeto de estudo que é suficientemente simples para ser estudado em detalhes e suficientemente abrangente para justificar um esforço significativo em transformar um sistema formal em uma ferramenta de trabalho que possa ser utilizada em aplicações reais.

## 1.3 Sistemas de Produção

A representação e manipulação do conhecimento por meio de sistemas de produção é uma das abordagens mais utilizadas na implementação de sistemas de processamento de conhecimento.

Sua grande aceitação decorre basicamente da razoável simplicidade de sua estrutura, bem como de seus bons resultados em aplicações práticas.

Um dos principais paradigmas que envolvem sistemas de processamento de conhecimento é o de que, em tais sistemas, deve haver uma clara distinção entre o conhecimento que está sendo processado, representado no que se chama de base de conhecimento, e o procedimento pelo qual este conhecimento é manipulado, chamado usualmente de máquina de inferência.

Em sistemas de produção, esta distinção é particularmente bem demarcada. A base de conhecimento é estruturada de modo a representar dois tipos básicos de elementos: os fatos e as regras. Estes elementos são então manipulados pela máquina de inferência, de maneira totalmente independente de seu conteúdo.

### 1.3.1 Base de Conhecimento

A base de conhecimento pode ser dividida em duas sub-bases distintas, que são complementares entre si.

A primeira delas, chamada de base de fatos, representa o conhecimento sobre fatos, ou seja, estados lógicos que podem ser fornecidos ou que podem ser inferidos pelo sistema. No primeiro caso, os fatos são levantados diante de uma base de dados, que pode ser alimentada pelo usuário ou mesmo por um processo. No segundo, os fatos inferidos pelo sistema são interpretados como a saída do sistema de processamento de conhecimento, servindo de para alimentar uma base de dados, responder perguntas do usuário ou mesmo servir para ativar determinadas ações de controle, no caso de processos.

Para o caso de sistemas de produção proposicionais, cada fato será representado por uma proposição, ou seja, um elemento simbólico que pode assumir valores verdadeiro ou falso. Este elemento estará relacionado a algum estado lógico do mundo que se deseja modelar. O sentido real da proposição, estará então ligado ao significado dos valores verdade que são fornecidos ao sistema, e dos valores verdade inferidos por ele, que serão então convertidos em saídas, fornecendo os resultados, que podem ser respostas ou ações de controle.

Alguns exemplos de proposições deste tipo seriam:

- peça_A_já_chegou_na_estação (1.1)
- peça_B_defeituosa (1.2)
- válvula_C_está_fechada (1.3)
- temperatura_esta_abaixo_do_limite (1.4)
- sistema_em_operação_crítica (1.5)
- enviar_peça_A_para_esteira (1.6)
- descartar_peça_B (1.7)
- abrir_válvula_C (1.8)

As proposições (1.1), (1.2), (1.3), (1.4) e (1.5) são do tipo conclusiva, ou seja, elas estão representando um estado lógico do sistema. Podem ser proposições de entrada (fornecidas ao sistema), intermediárias (em uma cadeia de raciocínio), ou mesmo de saída do sistema, quando o que se deseja é uma resposta. Por exemplo, no caso, a proposição (1.4), que avisa que a temperatura está abaixo do limite permitido, ou a (1.5) que fornece as condições de operação do sistema.

As proposições (1.6), (1.7) e (1.8), por outro lado, são proposições típicas de decisões, que serão transformadas posteriormente nas ações de controle correspondentes.

Todas estas proposições, são descritas em termos de seu significado prático, mas poderiam em princípio ser representadas por símbolos, por exemplo, A, B, X, Z, etc. Durante o processamento do conhecimento, o significado das mesmas não será considerado. Este significado só é importante nas entradas e saídas do sistema, quando existe efetivamente uma interface entre o mesmo e o mundo real. Esta independência entre os símbolos e seus significados é que permite a existência de máquinas de inferência totalmente desvinculadas da base de conhecimento.

A segunda sub-base é aquela que relaciona as diferentes proposições existentes na base de fatos, permitindo que a partir do conhecimento do estado lógico (valor verdade) de algumas proposições, possam ser inferidos os valores verdade de outras proposições, por um processo de raciocínio (inferência), que é exercido pela máquina de inferência, gerando os resultados que podem ser conclusões ou decisões.

O relacionamento entre os fatos (proposições) é dado na forma de regras. Regras são então um segundo elemento da base de conhecimento. Uma regra genérica é descrita como tendo uma premissa (antecedente), que pode ser composta, e uma conclusão (consequente). A operação de implicação é definida em termos formais, pela lógica matemática, e seu significado prático é o de que a veracidade do antecedente implica na veracidade do consequente. A implicação é representada por meio dos conectivos SE ... ENTÃO, que conectam os termos da seguinte forma:

SE <ANTECEDENTE> ENTÃO <CONSEQUENTE>

Os termos antecedente e consequente podem ser simples, quando são formados apenas por uma proposição em cada um deles, como por exemplo:

SE peça_B_defeituosa ENTÃO descartar_peça_B
SE a ENTÃO b

Ou então podem ser compostos, quando mais de uma proposição estão ligadas por conectivos lógicos do tipo "OU" e "E":

SE válvula_C_está_fechada E sistema_em_operação_crítica
ENTÃO abrir_válvula_C

SE ((a E b) OU (c E d)) E (f OU g) ENTÃO (h E i)

No tipo de regras mais usual, o antecedente é formado por diversas proposições ligadas por conectivos do tipo "E" (conjuntivo, gerando conjunções) e o consequente é formado por apenas uma proposição. Este tipo de regra pode ser colocado na forma de "cláusulas de Horn" (*Horn Clauses*). É um tipo de representação conveniente, visto que um conjunto de regras genéricas pode ser sempre reduzido a um conjunto de cláusulas de Horn, mantendo-se seu significado. As cláusulas de Horn são facilmente manipuláveis, necessitando de máquinas de inferência razoavelmente simples para processá-las. Um exemplo de base de regras gerada é colocado a seguir:

SE peça_A_já_chegou_na_estaçãoENTÃO enviar_peça_A_para_esteira
SE peça_B_defeituosa ENTÃO descartar_peça_B
SE válvula_C_está_fechada E temperatura_esta_abaixo_do_limite E
sistema_em_operação_crítica ENTÃO abrir_válvula_C
...

### 1.3.2 Máquina de Inferência

Em sistemas de produção, a máquina de inferência é o módulo que efetivamente processa o conhecimento, que estará previamente representado nas base de regras e base de fatos.

O paradigma básico que fundamenta as máquinas de inferência em sistemas de produção é baseado na regra de inferência "Modus Ponens". Este paradigma permite a associação de proposições com regras, gerando novas proposições, e pode ser colocado em termos básicos da seguinte forma:

$$\frac{\mathbf{A}\ ,\ \mathbf{A} \Rightarrow \mathbf{B}}{\mathbf{B}}$$

ou seja, assumindo-se que A é verdadeiro (proposição) e que A implica em B (regra), então deduz-se que B é verdadeiro (proposição).

Este paradigma é utilizado para a determinação dos valores-verdade desconhecidos das proposições da base de fatos. A utilização do mesmo pode

ser de maneira direta ou de maneira reversa. No modo direto, a partir do valor verdade do antecedente da regra, determina-se o valor-verdade do termo consequente. No modo reverso, assume-se a priori que o valor-verdade do consequente é "verdadeiro", e verifica-se então todos os termos do antecedente da regra, para garantir que tal suposição é correta. Caso todos os termos do antecedente tenham realmente o valor-verdade "verdadeiro", considera-se a suposição correta e determina-se então o valor-verdade do mesmo. Caso contrário, o valor-verdade do termo consequente permanece indeterminado.

A caracterização de uma máquina de inferência se inicia portanto na determinação do modo de aplicação do "modus ponens", se de modo direto ou de modo reverso.

As proposições que são citadas tanto nos antecedentes de algumas regras como nos consequentes, darão origem ao fenômeno do encadeamento. Este fenômeno ocorre, pois estando um fato com valor-verdade desconhecido, ele não permitirá o disparo (inferência de uma nova proposição por modus ponens) de uma regra que o tenha como componente do antecedente em sua primeira aplicação. Tendo seu valor-verdade definido por outra regra disparada, e que tenha o elemento no consequente, em uma aplicação posterior da regra anterior, a mesma já poderá disparar. Deste modo, não é a simples ordenação da aplicação da regra que se deve considerar, pois algumas regras deverão ser aplicadas novamente para que a inferência completa seja efetuada. Existem basicamente dois métodos básicos de raciocínio, cada um deles referente a um modo de aplicação do "Modus Ponens", ambos explorando o fenômeno do encadeamento.

O primeiro destes é chamado de encadeamento para frente *(forward chaining)* e corresponde à aplicação direta do "Modus Ponens". Como pode ocorrer o encadeamento de regras, as regras são aplicadas em sequência, repetidas vezes, gerando novos valores verdade a cada vez, até que não mais se modifiquem os valores verdade da base, após a aplicação de todas as regras (ou seja, nenhuma regra ainda não disparada dispara).

O segundo é chamado de encadeamento reverso *(backward chaining)*, correspondendo à aplicação reversa do "Modus Ponens". Neste, parte-se de uma proposição que deseja-se verificar, assumindo-se a priori seu valor verdade como verdadeiro. Efetua-se então a busca de maneira reversa, tentando-se validar esta asserção. Quando se chega a um antecedente de valor-verdade indeterminado, não se rejeita imediatamente a asserção original. Isto porque este pode estar encadeado em outra regra. O que se faz é tentar provar que também esta proposição deve ter seu valor-verdade em "verdadeiro", constituindo provisoriamente uma "nova meta". A busca (inferência) termina, tanto quando se valida a proposição, determinando-se que realmente o valor-verdade desta é "verdadeiro", como quando se esgotam as regras que não têm nenhuma proposição "pendurada" na lista de proposições que se deseja validar. Este método de inferência visa implementar uma faceta do raciocínio humano, onde busca-se encontrar a veracidade de uma hipótese baseada na veracidade dos elementos que sustentam esta hipótese. Ou seja, para que uma hipótese seja verdadeira é necessário que os elementos que a sustentam sejam verdadeiros. Confronta-se então estes elementos que deveriam ser verdadeiros com os valores verdade que se obtém dos fatos conhecidos. Caso o valor verdade

dos fatos conhecidos corresponda à expectativa de veracidade para sustentar-se a hipótese, a mesma é considerada verdadeira. Caso contrário assume-se que não existe evidência que possa confirmar sua veracidade.

Para o caso de encadeamento direto, a aplicação de todas as regras da base uma única vez a um conjunto de valores-verdade referentes às proposições da base de fatos é chamada de passo de inferência. A sucessiva aplicação de passos de inferência, até que a base não mais se modifique é chamada de ciclo de inferência.

### 1.3.3 Processamento Auxiliar

Normalmente, em sistemas de processamento de conhecimento, além do processamento realizado pela máquina de inferência, é necessário um processamento auxiliar. Isto ocorre, tanto na entrada como na saída do sistema, e sua finalidade básica é promover uma interface entre o usuário e o sistema. Principalmente em casos onde o sistema de processamento está acoplado a outros sub-sistemas, em um sistema de controle global, por exemplo, a entrada do sub-sistema de processamento de conhecimento será dada por uma base de dados comum, que é partilhada por ambos sub-sistemas. Estes outros sub-sistemas em princípio, não devem fornecer apropriadamente as informações necessárias (ou seja, o valor-verdade das proposições da base de fatos) e muito menos aproveitar os valores-verdade concluídos pelo sistema, de maneira imediata. Para suprir esta interface, um processamento auxiliar transforma as informações que são efetivamente fornecidas pelo sistema, em valores-verdade para as diferentes proposições utilizadas. Do mesmo modo, este colhe as decisões que foram tomadas pelo sistema, e as transforma efetivamente em sinais de controle que possam ser aproveitados pelo sub-sistema seguinte.

O processamento auxiliar será tanto mais elaborado quanto for o significado atribuído às proposições utilizadas na interface. Em casos mais simples, quando a interface é, por exemplo, o usuário respondendo a questões, este processamento poderá ser simplesmente colher a resposta e a converter em valores-verdade para uso interno. Em casos mais complicados, talvez seja necessário resolver uma relação (por exemplo: tempo_x_menor_que_o_sen(y)_2). Do mesmo modo, na saída, a interface talvez seja simplesmente imprimir os valores-verdade das proposições na tela de um computador. Ou então efetuar uma complicada ação de controle, como por exemplo uma proposição do tipo levar_carro_x_ate_posicao_y, que implicará em uma série de ações consecutivas de controle, como acelerar o carro, verificar a posição do mesmo e parar o carro quando atingir-se a posição correta.

De modo a promover a integração entre um sub-sistema de processamento de conhecimento a uma determinada aplicação, o processamento auxiliar torna-se de suma importância de modo a efetivar sua utilização com sucesso.

## 1.4 Aplicações em Ambientes de Tempo Real

O uso de sistemas de processamento de conhecimento tem sido proposto para as mais diversas áreas, desde a implementação de sistemas de diagnóstico médico, até para planejamento de produção, diagnóstico de falhas em placas de circuito impresso, prospecção de petróleo, aplicativos da área financeira, etc. Em todas estas aplicações, não é preponderante a consideração de restrições temporais durante o processamento. Para todos estes casos, o tempo que o sistema demora para tomar uma decisão não é de nenhum modo crítico para seu desempenho. Para o uso de tais sistemas em ambientes de tempo real, alguns cuidados fazem-se necessários, de modo a adequar algumas características aos requisitos impostos pelo aspecto de tempo real.

Vários casos de implementações bem sucedidas de tais sistemas são relatadas na literatura [16], sendo que alguns métodos são propostos de modo a contornar o aspecto de tempo real [15,19,32,39]. A maioria destes métodos são relativos a condições de "hard real-time", ou seja, na impossibilidade de se processar todo o conhecimento disponível diante de alguma restrição temporal, tais métodos orientam na escolha de regras, selecionando as que devem ser utilizadas primeiro, de modo que as demais possam ser descartadas, caso não haja tempo para processá-las.

Laffey et.al. [16] levanta uma série de problemas que se encontram quando se utilizam os atuais *shells* existentes no mercado, para a implementação em ambientes de tempo real:

- Não são rápidos o suficiente para as aplicações.
- Não possuem capacidade (ou têm pouca) de realizar inferências levando em conta o aspecto temporal.
- São de difícil integração com os softwares convencionais.
- Possuem nenhuma ou pouca facilidades em focalizar a atenção em eventos significativos.
- Não possuem capacidade de integração com o relógio de tempo real.
- Não estão capacitados para manusear eventos assíncronos.
- Não estão capacitados para manusear interrupções de hardware e software.
- Não possuem integração eficiente para aquisição de sinais que não a intervenção humana.
- Não existem métodos para validar os próprios shells ou a base de conhecimento que estes manipulam.
- Não garantem o tempo de resposta.
- Só funcionam em hardwares não habilitados a funcionar em ambientes ruidosos, típicos da aplicação industrial.

Estas dificuldades permitem que se faça o levantamento de alguns requisitos para a construção de sistemas de processamento de conhecimento quando os mesmos têm como destino a aplicação em ambientes de tempo real:

- **Integração eficiente do processamento numérico-simbólico:**

Este requisito diz respeito à capacidade de interfaceamento dos sistemas de processamento de conhecimento (SPC) com o mundo real. Para tal, deve-se ter um processamento auxiliar (conforme a seção 1.3.3) eficiente, de modo que o sistema não tenha como gargalo a entrada destes dados, e do mesmo modo, permita ao engenheiro do conhecimento uma certa flexibilidade na elaboração das regras, diante da integração que as mesmas devem ter com o mundo real.

- **Operação Contínua:**

Este requisito exige que o sistema continue em funcionamento, mesmo diante de uma eventual pane parcial nas entradas de sinal, mantendo sua integridade e gerando as melhores respostas possíveis diante das entradas que continuam a funcionar. Este requisito pode ser satisfeito por exemplo, com o emprego de alguma lógica de *default* [29], que verifique a integridade dos dados e substitua os dados não confiáveis por seus *defaults*, ou que detecte que não há condições de se continuar operando (por falta de informações) e invoque alguma rotina de emergência.

- **Mecanismo de foco de atenção:**

Este requisito evita que em sistemas com bases de regras muito grandes, todas as regras sejam continuamente aplicadas, gerando um *overhead* muito grande que impossibilite a geração da resposta dentro dos limites de tempo. Para implementar este requisito, utiliza-se a estruturação da base de regras, normalmente controladas por uma base de meta-regras, que tenta levantar padrões de comportamento do sistema, e direciona o sistema para sub-base de regras específicas para abordar tais comportamentos.

- **Serviços de gerenciamento de interrupções:**

Este requisito pede que o sistema possa interpretar eventos assíncronos, que podem ter maior prioridade. Principalmente quando se utiliza bases de regras estruturadas, é necessário que o sistema possa ser redirecionado, diante da ocorrência de um evento que modifique, por exemplo, o padrão de comportamento do sistema, e possa selecionar então uma nova base de regras de modo a satisfazer as condições do novo padrão de comportamento do sistema.

- **Utilização ótima do ambiente:**

Os SPC utilizados em tempo real, em geral não necessitarão de serviços que existem nos *shells* convencionais, tais como os módulos explicativos (que explicam uma decisão) e outros mais que não levam em conta o aspecto de tempo real e por isso podem ser grandes consumidores de tempo de processamento. Uma máquina de inferência para tempo real deve ser a mais otimizada possível de modo a processar somente a informação necessária para o bom desempenho do sistema.

- **Garantia do Tempo de Resposta:**

Um requisito importante em sistemas de tempo real, é que o SPC possa garantir a resposta para uma entrada genérica, de modo que se possa garantir que a resposta do sistema não chegará, por exemplo, depois que a mesma já não é mais aplicável. A simples possibilidade de que este tempo de resposta seja mensurável, permite o projeto adequado, alterando-se o hardware (ou mesmo otimizando a base de conhecimento) para que a resposta esteja sempre dentro dos limites colocados pela aplicação.

- **Capacidade de Processamento de Dados Temporais:**

Este requisito existe de modo que o engenheiro do conhecimento possa ter acesso a dados históricos, de modo a utilizá-los na elaboração das regras. Este serviço será em princípio de responsabilidade do processamento auxiliar (seção 1.3.3), que deverá prever a capacidade de acesso, manutenção e análise estatística destes dados históricos.

- **Manutenção da validade dos dados:**

Este requisito lembra que os dados utilizados em sistema de tempo real são normalmente de origem dinâmica, podendo estar constantemente se modificando. Para permitir o processamento de dados deste tipo, é necessário, ou que se tenha um método de inferência não-monotônico ou que o sistema tenha um tempo de resposta tão pequeno que se garanta que neste intervalo de tempo os dados não serão modificados e portanto o sistema será monotônico neste intervalo, e as bases de fatos são então atualizadas a cada ciclo de inferência.

Laffey conclui, ao final de uma análise de diversas implementações efetuadas, que na maioria destas, o desempenho é profundamente prejudicado devido ao fato de se utilizarem métodos tradicionais para uma aplicação para a qual eles não eram adequados. Segundo ele, existe uma necessidade premente da elaboração de máquinas de inferência mais eficientes, que possam processar o conhecimento de modo mais eficaz e produtivo, tendo-se em conta o aspecto de tempo real. Alguns algoritmos são relatados como tendo alta eficiência, tais como o RETE [11] e o EUREKA II [12].

## 1.5 Representação Estruturada do Conhecimento

Como foi visto na seção anterior, um dos quesitos importantes na elaboração de um SPC direcionado para aplicações de tempo real é a capacidade de foco de atenção. Esta capacidade, pode ser implementada efetuando-se uma representação estruturada do conhecimento [2]. Normalmente o conhecimento do especialista está estruturado, de modo que o mesmo toma decisões não só diante do estado geral do sistema, mas este estado é estruturado em diversas situações diferentes, gerando cenários sobre os quais o especialista passa a tomar suas decisões. Esta representação estruturada pode ser conveniente de diversas formas. Em uma primeira instância, dividindo-se a base de conhecimento em diversas sub-bases, permite-se uma melhor aquisição do conhecimento por parte do especialista, que pode explicar suas decisões diante de cenários definidos. De outro modo também é interessante, pois o sistema não precisa gastar um tempo inútil no processamento de regras que não se encaixam no cenário atual do comportamento do sistema controlado, aumentando nitidamente o desempenho do SPC.

Fundamentalmente um SPC que utilize uma representação estruturada do conhecimento deve ter como requisito, a facilidade no chaveamento entre diversas bases de regras. Deste modo, o sistema começa vasculhando a base de meta-regras, e quando algum padrão de comportamento for detectado, chaveia-se para a sub-base específica que tem as regras que tratam este comportamento, gerando decisões adequadas ao cenário em questão. O chaveamento de volta para as meta-regras pode ocorrer periodicamente, para a garantia de que ainda se encontra sobre as condições que determinam o padrão de comportamento, ou pode ser forçado diante de qualquer evento externo que ocorra, como uma interrupção, por exemplo, Isto garante não só a capacidade de foco de atenção mas também garante o tratamento de eventos assíncronos. Processando menos regras, um SPC assim implementado também torna-se mais eficiente, melhorando seu tempo de resposta.

## 1.6 Proposta de Tese

Conforme observou-se nas seções introdutórias, é grande o interesse por esquemas de representação e processamento de conhecimento, diante da hipótese da aplicação de tais sistemas na resolução de problemas que seriam considerados muito complicados para serem resolvidos pelos métodos convencionais da teoria de sistemas.

Do mesmo modo, a maioria dos sistemas de processamento de conhecimento é ainda hoje implementada utilizando-se procedimentos de inferência de âmbito geral, ou seja, procedimentos em que não existe uma

preocupação mais explícita com todos os requisitos levantados na seção 1.4, característicos de ambientes de tempo real.

Considerando que as aplicações em controle em tempo real e automação industrial são um conjunto importante de aplicações em que os sistemas de processamento de conhecimento podem suprir as dificuldades encontradas diante da teoria de sistemas convencional, e que a eficiência de execução de uma máquina de inferência pode ser vital para sua utilização nesta classe de aplicações, propõe-se neste trabalho o desenvolvimento de um procedimento de inferência voltado para as particularidades de sistemas em tempo real, quanto a necessidade e urgência de informações, para processar o conhecimento de uma forma altamente eficaz, de modo que o mesmo possa ser efetivamente utilizado diante das restrições impostas pelos sistemas em tempo real.

Looney [21,22] propôs um método matricial de inferência para sistemas de produção, que devido ao seu caráter matricial mostrou-se interessante, diante da possibilidade da paralelização de suas operações, e mesmo da possibilidade de se garantir um tempo de resposta de ordem quadrática, ao contrário dos algoritmos convencionais que tem um tempo de resposta normalmente exponencial. Entretanto, as regras consideradas por ele não permitem de maneira adequada a utilização de implicações, ou seja, não implementa de modo adequado a conjunção de elementos nos antecedentes das regras.

Warfield [37] afirma que a utilização de matrizes binárias é particularmente interessante pois podem representar a existência ou inexistência de um determinado elemento, que em casos de sistemas de produção pode corresponder a existência ou inexistência de um determinado símbolo em uma regra. Com isso, a representação do conhecimento por meio de matrizes binárias pode ser particularmente interessante em SPC's voltados para aplicações em tempo real, diante do que hoje se conhece sobre análise matricial, permitindo uma representação adequada e apropriada para esta classe de aplicações.

Neste trabalho, será desenvolvido um procedimento de inferência que procura explorar o caráter da representação matricial, para uma implementação eficiente de SPC's em ambientes de tempo real, tais como a representação estruturada do conhecimento, a eficiência do processamento do conhecimento, e se aproveitar das possibilidades de paralelismo colocadas pela representação matricial. Espera-se obter um procedimento que seja mais apropriado, portanto, que os procedimentos convencionais, no processamento de conhecimento levando-se em conta sua utilização em ambientes de tempo real.

Tanto o procedimento de inferência quanto os esquemas de representação de conhecimento são enfocados sob o ponto de vista da lógica proposicional booleana e da lógica nebulosa. É elaborado um estudo comparativo, para o caso de baseado na lógica proposicional convencional, entre o procedimento aqui proposto e aqueles tradicionais abordados na literatura. Em adição, é feita uma aplicação a um problema de alocação em sistemas de transporte vertical.

## 1.7 Resumo

Neste capítulo discorreu-se sobre conceitos básicos a respeito de sistemas de processamento de conhecimento. Considerou-se o problema da representação do conhecimento, dando-se um enfoque maior sobre a lógica matemática, e na estrutura de sistemas de produção. Definiu-se tanto o significado de inferência como a estrutura básica de um sistema de processamento de conhecimento, ou seja, a divisão entre o conhecimento e a operação sobre ele. Apresentou-se os dois métodos básicos de inferência (encadeamento direto e reverso) e os tipos de processamento auxiliar necessário.

Em seguida, colocou-se alguns requisitos para o uso de SPC em ambientes de tempo real, bem como levantou-se os problemas existentes atualmente nesta área.

Seguiu-se a colocação da proposta do trabalho, diante de sua motivação básica e dos trabalhos já existentes, dentro do mesmo contexto..

No capítulo 2, será desenvolvido o procedimento de inferência proposto, de modo detalhado, fundamentando-se sua concepção dentro da teoria da lógica proposicional clássica.

# 2. UM PROCEDIMENTO DE INFERÊNCIA

## 2.1 Introdução

Este capítulo apresenta um procedimento de inferência para sistemas de produção voltados para aplicações em tempo real.

Baseado em lógica proposicional e utilizando encadeamento para frente, o mesmo fundamenta-se em operadores matriciais booleanos, permitindo uma implementação computacional eficiente, incluindo facilidades de paralelização. Além disso, sua estrutura praticamente não é modificada quando estendemos seu funcionamento para sistemas de produção nebulosos ("fuzzy systems"), tornando-o uma alternativa atraente para implementação de sistemas de controle baseados em regras.

## 2.2 Descrição do Método

### 2.2.1 Codificação da Base de Regras

A base de regras utilizada pelo procedimento será descrita por duas matrizes -operacionais C e D. A matriz C, utilizada na forma negada ($\overline{C}$), é chamada de matriz conjuntiva pois, utilizada com o operador matricial disjuntivo, proporciona a conjunção das entidades citadas nos antecedentes das regras. A matriz D, chamada de matriz disjuntiva, utiliza o operador matricial conjuntivo, realizando a disjunção entre os consequentes, decorrentes da ativação das diversas regras.

O par formado pelas matrizes C e D constitui uma Base de Regras Operacional, visto que a estrutura destas duas matrizes contém informações sobre as regras que serão utilizadas no processo de inferência. O processo de inferência será decorrente do uso das matrizes, dos operadores matriciais e dos vetores de fatos.

DEFINIÇÃO 2.1: Regra Proposicional

Seja E o conjunto das entidades que representam os símbolos proposicionais sobre um universo de discurso.

$$E = \left\{ e_i \mid e_i \text{ símbolo proposicional}, \quad 1 \leq i \leq n \right\}$$

Define-se uma Regra R pela união de suas estruturas sintática e semântica.

A estrutura sintática de uma regra é definida como uma sequência de palavras, cada qual formada por um conjunto de caracteres, do modo a seguir:

SE antecedente ENTAO consequente

onde antecedente corresponde a um conjunto de 1 a n-1 elementos de E, relacionados conjuntivamente pelo operador booleano $\wedge$ ("and" ou "E"), e consequente corresponde a um elemento de E.

Exemplo 2.1:

$$\text{Seja } E = \{e_1, e_2, \ldots, e_{10}\}$$

Exemplos de antecedente

- $e_1$ E $e_2$ E $e_5$
- $e_1 \wedge e_2 \wedge e_5$
- $e_7$ E $e_9$
- $e_3$

Exemplos de consequente

- $e_1$
- $e_7$
- $e_4$

Exemplo de Regra Completa

- SE e1 E e4 ENTAO e2

Seja A um subconjunto de E, tal que os elementos de A sejam membros do antecedente da estrutura sintática da regra R:

$$A = \left\{ a_i \mid a_i \in E, \text{ai membro de antecedente}, \quad 1 \leq i \leq k, \; k < n \right\}$$

Seja Q um subconjunto unitário de E, tal que seu elemento q corresponde ao consequente da estrutura sintática da regra R:

$$Q = \left\{ q \mid q \in E, q \text{ é consequente}, \; 1 \leq i \leq n \right\}$$

A estrutura semântica de uma regra é definida como uma sentença em lógica proposicional, do seguinto tipo:

$$\bigwedge_i \ (a_i) \Rightarrow q$$

onde cada $a_i$ é um elemento de A, q um elemento de Q , " $\bigwedge$ " o operador lógico conjuntivo e "⇒" o operador lógico implicativo da lógica proposicional.

Isto significa que a veracidade de todos os elementos de A (antecedentes da regra) implica na veracidade de Q (consequente da regra).

DEFINIÇÃO 2.2: Base de Regras

Uma base de regras BR é um conjunto de regras $R_j$ onde $1 \leq j \leq m$, (m é o número de regras da base ).

$$BR = \left\{ R_j \mid R_j \text{ é uma regra, } 1 \leq j \leq m \right\}$$

DEFINIÇÃO 2.3 : Conteúdo Semântico

Seja E o universo de discurso.

Define-se o conteúdo semântico de uma regra R , como sendo uma função $\varphi$ que mapeia todas as interpretações possíveis para E ($I_E$) , em {falso,verdadeiro}, por meio da sentença lógica correspondente à estrutura semântica da regra R .

Ou seja, para $Aj \subset E$ e $Q_j \subset E$, se a estrutura semântica de R é:

$$(a_{j1} \wedge a_{j2} \wedge \ldots \wedge a_{jk}) \Rightarrow q_j$$

então:

$$\varphi(R_j, I_E) = ((a_{j1} \wedge a_{j2} \wedge \ldots \wedge a_{jk}) \Rightarrow q_j)$$

O conteúdo semântico de uma base de regras pode ser definido de maneira semelhante, para um conjunto de regras BR, onde agora mapeia-se toda as interpretações possíveis de E, em {falso, verdadeiro}, por meio da sentença lógica correspondente à união dos conteúdos semânticos de cada regra de BR, ou seja:

$$\varphi(BR, I_E) = \varphi(R_1, I_E) \vee \varphi(R_2, I_E) \vee \ldots \vee \varphi(R_m, I_E)$$

que de certo modo relaciona a uma base de regras um significado no mundo real.

DEFINIÇÃO 2.4 : Codificação de uma Base de Regras

Entende-se por codificação de uma regra, uma alteração na estrutura sintática da mesma sem alterar, no entanto, seu significado.

Deste modo, a codificação de uma base de regras é apenas uma mudança na representação do conjunto de regras, sendo preservado seu conteúdo semântico.

Seja BR uma base de regras descrita conforme a definição 2.2.

Sejam C (m × n) e D (n × m), respectivamente as matrizes conjuntiva e disjuntiva

Define-se então uma codificação da base de regras BR como o seguinte mapeamento:

$$Cod(BR,\{C,D\}) : BR \rightarrow \{C,D\}, \varphi(BR,I_E)=\varphi(\{C,D\},I_E), \forall I_E$$

Uma descrição mais aprofundada da estrutura das matrizes C e D, e maiores detalhes sobre como é feito este mapeamento serão apresentados nas seções 2.3 e 2.4.

Operacionalmente, o procedimento de inferência utilizará a representação das matrizes C e D para a manipulação do conhecimento. Entretanto, o conteúdo semântico das regras permanece o mesmo. Esta equivalência ocorre pois, a nível de interfaceamento homem-máquina, a representação colocada pela definição 2.2 é mais amigável. Já para a manipulação em tempo real, pelo procedimento de inferência, a representação em termos das matrizes C e D proporciona um melhor desempenho computacional.

### 2.2.2 Codificação da Base de Fatos

Para o procedimento de inferência em questão, serão consideradas duas bases de fatos. A primeira, chamada de base de fatos de entrada, contém os fatos que são assumidos verdadeiros "à priori", antes do processo de inferência ser realizado. A segunda, chamada de base de fatos inferidos, indicará os fatos que são verdadeiros após o processo de inferência ser realizado. Esta base deverá conter tanto os fatos verdadeiros iniciais (contidos na base de fatos de entrada), quanto os fatos que forem inferidos pelo procedimento .

Cada uma das bases será representada por um vetor cujos elementos são variáveis booleanas. Este vetor terá tantos elementos quantos forem as entidades citadas na base de regras, havendo uma relação bi-unívoca entre cada entidade e um índice do vetor. Assim, se o elemento i do vetor assumir valor 1, isso significará que a entidade correspondente ao índice i será verdadeira. Do mesmo modo, se assumir o valor 0, a entidade será falsa. Estes vetores serão conhecidos como vetores de fatos.

DEFINIÇÃO 2.5: Vetor de Fatos

Seja E (como antes) o conjunto de todos as entidades presentes nas regras que compõem a base de regras

Seja Card o operador que define a cardinalidade de um conjunto, ou seja, para o conjunto E:

$$Card(E) = n$$

Seja x um vetor tal que:

$$x = \left\{ x_i \mid x_i \in \{ 0,1 \},\ 1 \leq i \leq Card(E) \right\}$$

Sendo as entidades de uma base de fatos BF um sub-conjunto de E, a mesma poderá ser representada a partir de um vetor x, da seguinte forma:

$$x_k = \begin{cases} 1 \text{ , se } e_k \in BF \\ 0 \text{ , caso contrário} \end{cases} \qquad \forall k \in [1,n]$$

Neste caso considerar-se-á que o vetor x representa a base de fatos BF, e o mesmo será chamado de um vetor de fatos. As bases de fatos serão codificadas por meio de vetores de fatos, sendo que a base de fatos de entrada passará a ser descrita pelo vetor xe, vetor de fatos de entrada, e a base de fatos inferidos, descrita pelo vetor xs, vetor de fatos inferidos.

### 2.2.3 Passo de Inferência

Durante o processo de inferência, faz-se necessário definir uma etapa que corresponde à aplicação de todas as regras de uma base de regras a uma base de fatos, incorporando a ela as entidades obtidas pelas regras aplicadas. Este processo é definido como sendo o passo de inferência.

DEFINIÇÃO 2.6 : Passo de Inferência

Sejam C e D as matrizes conjuntiva e disjuntiva correspondentes à base de regras operacional.

Sejam $\dot{\wedge}$ e $\dot{\vee}$ os operadores matriciais conjuntivo e disjuntivo.

Seja a operação de negação (Neg(A) = $\bar{A}$) correspondente à negação booleana elemento a elemento da matriz A.

Seja a operação de soma vetorial ("x + y") a soma booleana elemento a elemento dos vetores x e y.

Sejam xi e xf dois vetores de fatos.
Define-se um passo de inferência por:

$$xf = \text{PassoInferencia}(xi) = xi + D \dot{\wedge} (\overline{C} \dot{\vee} xi)$$

### 2.2.4 Ciclo de Inferência

O ciclo de inferência é necessário a fim de proporcionar o encadeamento entre as regras, quando este existir. Na realidade um ciclo de inferência não passa da sucessiva aplicação de passos de inferência a um vetor de fatos, até que o mesmo não se modifique após um número de passos realizados.

DEFINIÇÃO 2.7 : Ciclo de Inferência

O ciclo de inferência é definido como sendo um conjunto finito de aplicações sucessivas de um passo de inferência a um vetor de fatos.

Sejam xe o vetor de fatos de entrada, xf o vetor de fatos de saída e x um vetor de fatos auxiliar. Seja o operador atribuição vetorial (": = (a,b) ≡ a : = b") definido como a atribuição elemento a elemento do vetor b ao vetor a.Seja o operador diferente (" ≠" ) definido da seguinte forma:

$$\neq(x,y) \equiv x \neq y; x, y \text{ vetores de fatos, } = \begin{cases} \text{falso,} & \text{se } x_i = y_i \quad i \in [1,n] \\ \text{verdadeiro,} & \text{caso contrário} \end{cases}$$

```
procedimento CICLO_DE_INFERÊNCIA
INICIO
    xf : = xe;
    ENQUANTO (xf ≠ x)
    INICIO
        x : = xf;
        xf : = PassoInferencia(x);
    FINAL
FINAL
```

O ciclo de inferência se confunde com o próprio procedimento de inferência, visto que após sua execução obtém-se uma inferência completa a partir de um dado vetor de fatos de entrada, tomando-se o vetor de fatos inferidos como resultado.

## 2.3 Estrutura da Matriz C

Definiu-se como base de regras operacional o par de matrizes C e D. Nesta seção, define-se a estrutura da matriz C, a matriz conjuntiva.

Na estrutura da matriz, cada linha da mesma corresponde à lista de antecedentes de uma dada regra da base de regras, e cada coluna uma determinada entidade, de maneira análoga a do vetor de fatos. Assim a matriz C tem dimensões m × n, onde m é o número de regras contidas na base de regras e n é o número de entidades presentes na base de regras.

DEFINIÇÃO 2.8 : Estrutura da Matriz C

Seja E o conjunto das entidades do universo de discurso.
Seja BR uma base de regras.
Seja $R_i$ a i-ésima regra de BR.
Seja $A_i$ o conjunto de entidades que são membros do antecedente da regra $R_i$.
Define-se a estrutura da matriz C do seguinte modo:

$$C = \left\{ c_{ij} \mid c_{ij} = f(i,j), 1 \leq i \leq m, 1 \leq j \leq n \right\}$$

$$f(i,j) = \begin{cases} 1, & \text{se } e_j \in A_i \\ 0, & \text{caso contrário} \end{cases}$$

## 2.4 Estrutura da Matriz D

Nesta seção define-se a estrutura da matriz D, a matriz disjuntiva.

Esta matriz, deve efetuar a disjunção das entidades citadas como consequentes nas diversas regras aplicadas. Sendo assim, a i-ésima linha da matriz deve representar, de modo semelhante ao de um vetor de fatos, as regras que tem como consequente a entidade i. Portanto a matriz D tem dimensões n × m, onde n é o número de entidades presentes na base de regra e m é o numero de regras contidas na base de regras.

DEFINIÇÃO 2.9 : Estrutura da Matriz D

Seja E o conjunto de entidades do universo de discurso.
Seja BR uma base de regras.
Seja $R_j$ a j-ésima regra de BR.

Seja $Q_j$ o conjunto de entidades que são membros do consequente da regra $R_j$.

Define-se então a estrutura da matriz D como sendo:

$$D = \left\{ d_{ij} \mid d_{ij} = g(i,j), 1 \leq i \leq n, 1 \leq j \leq m \right\}$$

$$g(i,j) = \begin{cases} 1, & \text{se } e_i \in Q_j \\ 0, & \text{caso contrário} \end{cases}$$

## 2.5 Operador Matricial Conjuntivo

O operador matricial conjuntivo é o operador pelo qual se faz a disjunção dos consequentes das diversas regras aplicadas. Sua definição se dá de modo semelhante ao de um operador matricial multiplicativo, trocando-se os operadores escalares aritméticos por operadores escalares booleanos.

DEFINIÇÃO 2.10 : Operador Matricial Conjuntivo

Seja "$\wedge$" o operador escalar booleano conjuntivo ("and" ou "e").
Seja "$\vee$" o operador escalar booleano disjuntivo ("or" ou "ou").
Seja M uma matriz de dimensões n × m com elementos $m_{ij}$ booleanos.
Seja x um vetor de elementos booleanos de dimensão m.
Seja y um vetor de elementos booleanos de dimensão n.
Define-se o operador matricial conjuntivo "$\dot{\wedge}$" da seguinte maneira:

$$y = \dot{\wedge}(M,x) = M \,\dot{\wedge}\, x = \left\{ y_i \mid y_i = \bigvee_j (m_{ij} \wedge x_j) \right\}$$

onde $\bigvee_j$ é a somatória booleana: $\bigvee_j x_j = x_1 \vee x_2 \vee \ldots \vee x_m$.

## 2.6 Operador Matricial Disjuntivo

O operador matricial disjuntivo é o operador pelo qual se faz a conjunção dos antecedentes das regras. Seu funcionamento é o dual do operador matricial conjuntivo, ou seja, toma-se a mesma definição trocando-se os operadores escalares booleanos "$\wedge$" por "$\vee$" e vice-versa.

DEFINIÇÃO 2.11 : Operador Matricial Disjuntivo

Seja "$\wedge$" o operador escalar booleano conjuntivo ("and" ou "e").
Seja "$\vee$" o operador escalar booleano disjuntivo ("or" ou "ou").
Seja M uma matriz de dimensões m × n com elementos $m_{ij}$ booleanos.
Seja x um vetor de elementos booleanos de dimensão n.
Seja y um vetor de elementos booleanos de dimensão m.
Define-se o operador matricial disjuntivo "$\dot{\vee}$" da seguinte maneira:

$$y = \dot{\vee}(M,x) = M \,\dot{\vee}\, x = \left\{ y_i \mid y_i = \bigwedge_j (m_{ij} \vee x_j) \right\}$$

onde $\bigwedge_j$ é a produtória booleana: $\bigwedge_j x_j = x_1 \wedge x_2 \wedge \ldots \wedge x_n$.

## 2.7 Procedimento de Inferência

Nesta seção será apresentado o mecanismo que envolve as matrizes C e D, os vetores de fatos e os operadores matriciais, resultando no procedimento de inferência.

TEOREMA 2.1 : Teorema do Passo de Inferência

Seja BR uma base de regras.
Sejam C e D a codificação da base de regras BR em termos das matrizes conjuntiva e disjuntiva.
Sejam "$\dot{\wedge}$" e "$\dot{\vee}$" os operadores matriciais conjuntivo e disjuntivo.
Seja "+" o operador vetorial de soma booleana.
Sejam xe o vetor de fatos de entrada e xi o vetor de fatos inferidos.
Deste modo, a partir de um vetor de fatos de entrada genérico xe, e aplicando-se a fórmula

$$xf = xe + D \,\dot{\wedge}\, (\overline{C} \,\dot{\vee}\, xe)$$

obtém-se em xf os valores verdade de todos os fatos que podem ser inferidos por meio da aplicação das regras contidas em BR sobre os fatos representados por xe, incluindo os fatos de entrada.

**PROVA:**

A prova deste teorema é dividida em duas partes. Na primeira, provaremos que a operação $(\overline{C} \,\dot{\vee}\, xe)$ leva a um vetor intermediário y, contendo o valor verdade do antecedente de cada regra. Na segunda parte, prova-se que a partir do valor verdade do antecedente de cada regra e da operação $D \,\dot{\wedge}\, y$,

obtém-se, implicitamente, por "modus ponens" o valor verdade de todos as entidades pertencentes aos consequentes das regras e com isso todos os fatos que podem ser inferidos a partir dos fatos de entrada. A soma vetorial booleana deste resultado com o vetor de fatos de entrada faz com que o vetor de fatos inferidos contenha o valor verdade de todos as entidades do universo do discurso, após a aplicação das regras da base de regras BR.

**Parte 1:**

Pela definição do operador matricial disjuntivo, temos que

$$y = \overline{C} \dot{\vee} x = \left\{ y_i \mid y_i = \bigwedge_j (\overline{c}_{ij} \vee x_j) \right\}$$

O valor de y passa então a ser a produtória booleana

$$y = (\overline{c}_{i1} \vee x_1) \wedge (\overline{c}\, i_2 \vee x_{2)} \wedge \ldots \wedge (\overline{c}\, i_n \vee x_n)$$

Pela definição da estrutura da matriz C, temos que $c_{ij} = 1$ se a entidade $e_j$ pertence ao antecedente da regra $R_i$, sendo 0 caso contrário. Se $c_{ij} = 0$, então $\overline{c}_{ij} = 1$. Para estes casos, a parcela$(\overline{c}_{ij} \vee x_j)$ será 1, qualquer que seja o valor de $x_j$. Deste modo, o valor desta parcela não irá influir no valor do produtório. Supondo então um conjunto $K^C_i$, formado pelos índices k tais que $\{\overline{c}_{ik} = 1$, e um conjunto $L^C_i$ formado pelos índices l tais que $\overline{c}_{il} = 0$, podemos então dividir o produtório em duas partes:

$$y_i = \left( \bigwedge_k (\overline{c}_{ik} \vee x_k) \right) \wedge \left( \bigwedge_l (\overline{c}_{il} \vee x_l) \right)$$

onde $\overline{c}_{ik}$ é sempre $1\ \forall k \in K^C_i$ e $\overline{c}_{il}$ é sempre $0\ \forall l \in L^C_i$.

Como $(\overline{c}_{ik} \vee x_k)$ é sempre igual a 1, $\forall k \in K^C_i$, temos que

$$\left( \bigwedge_k (\overline{c}_{ik} \vee x_k) \right) = 1 \quad \text{,e então}$$

$$y_i = \left( \bigwedge_l (\overline{c}_{il} \vee x_l) \right)$$

Do mesmo modo, como $\overline{c}_{il}$ é sempre igual a 0, $\forall l \in L^C_i$, temos que:

$$y_i = \bigwedge_l x_l$$

Deve-se lembrar que se $\bar{c}_{il} = 0$, ($c_{il} = 1$) então $e_l$ pertence ao antecedente da regra $R_i$, $\forall l \in L^C_i$ . Do mesmo modo, o valor verdade de $y_i$ só será 1 se todos os valores de $x_l$ forem 1. Como todos as entidades $e_l$ pertencem ao antecedente da regra R isto significa que a conjunção de todos $x_l$ é equivalente à conjunção de todos os valores verdades das entidades que são membros do antecedente da regra $R_i$ e com isso o valor de $y_i$ é equivalente ao valor verdade do antecedente da regra $R_i$, c.q.d.

**Parte 2:**

Pela definição do operador matricial conjuntivo, temos:

$$xf = D \dot{\wedge} y = \left\{ xf_i \mid xf_i = \bigvee_j ( d_{ij} \wedge y_j ) \right\}$$

Pela definição da estrutura da matriz D, temos que $d_{ij} = 1$ se $e_i$ pertence ao consequente da regra $R_j$ e 0 em caso contrário. Supondo então um conjunto $K^D_i$, formado pelos índices k tais que $d_{ik} = 1$, e um conjunto $L^D_i$, formado pelos índices l tais que $d_{il} = 0$, podemos então dividir a somatória booleana em duas partes:

$$xf_i = \left( \bigvee_k ( d_{ik} \wedge y_k ) \right) \vee \left( \bigvee_l ( d_{il} \wedge y_l ) \right)$$

onde $d_{ik}$ é sempre 1, $\forall k \in K^D_i$ e $d_{il}$ é sempre 0, , $\forall l \in L^D_i$.

Se $d_{il}$ é sempre 0, a parcela $( d_{il} \wedge y_l )$ será sempre 0, qualquer que seja o valor de $y_l$, $\forall l \in L^D_i$.

Com isso, temos que

$$\bigvee_l ( d_{il} \wedge y_l ) = 0$$

e com isso

$$xf_i = \left( \bigvee_k ( d_{ik} \wedge y_k ) \right)$$

Como $d_{ik}$ também é sempre igual a 1, $\forall k \in K^D_i$, temos que:

$$xf_i = \bigvee_k y_k \quad (2.1)$$

**OBS. : Modus Ponens**

Pela regra de inferência "Modus Ponens" temos:

$$\frac{A \ , \ A \Rightarrow B}{B}$$

ou seja, tendo que A é verdade e que A implica B, deduzimos que B é verdade.

Lembrando que para $d_{ik} = 1$, temos que $\forall k \in K^D_i$, $e_i \in Q_k$ (onde $Q_k$ guarda o consequente da regra k), o que significa que $e_i$ é consequente da regra $R_k$.

Deste modo, o conteúdo semântico da regra $R_k$ pode ser re-escrito da seguinte forma:

$$y_k \Rightarrow e_i$$

Deste modo temos que para $\forall k \in K^D_i$, a expressão $y_k \Rightarrow e_i$ assume o valor verdadeiro, ou seja a fórmula é válida.

Com isso, temos que, por "modus ponens":

$$\frac{y_k \ , \ y_k \Rightarrow e_i}{e_i}$$

para todos os valores de k pertencentes a $K^D_i$. Portanto $e_i$ pode ser deduzido a partir da regra k, se $y_k = 1$.

Logo, sendo $y_k = 1$, $\forall k \in K^D_i$, teremos por definição $xf_i = 1$. Consequentemente

$$xf_i = \bigvee_k y_k$$

Que é equivalente a (2.1).

Deste modo, provamos a segunda parte do teorema e o mesmo por inteiro.

## TEOREMA 2.2 : Teorema do Ciclo de Inferência

Seja BR uma base de regras.

Sejam C e D a representação de BR por meio de suas matrizes conjuntiva e disjuntiva.

Sejam "$\dot{\wedge}$" e "$\dot{\vee}$" os operadores matriciais conjuntivo e disjuntivo.

Seja "+" o operador vetorial de soma booleana.

Sejam xe o vetor de fatos de entrada e xi o vetor de fatos inferidos.Seja P o operador equivalente ao passo de inferência

$$P(xe) = xe + D \dot{\wedge} (\overline{C} \dot{\vee} xe)$$

A aplicação de P sucessivas vezes a xe, tomando a cada vez a saída da operação como entrada da operação consecutiva, resultará na inferência completa a partir da base de regras BR sobre uma interpretação genérica representada em xe.

**PROVA:**

Pelo teorema 2.1, temos que o passo de inferência equivale à aplicação de todas as regras de uma base BR sobre um vetor de fatos genérico. Deste modo, a aplicação sucessiva do operador P sobre um vetor de fatos xe, tomando a saída de cada operação como a entrada da operação consecutiva, é equivalente à aplicação sucessiva de todas as regras de uma base de regras sobre uma base de fatos. Supondo que as regras possam se encadear (ou seja, o consequente de uma regra pode ser o antecedente de outra regra), formando uma cadeia de g níveis, temos que após a aplicação de g vezes o operador P, teremos aplicado todas as regras que poderiam ser aplicadas, tendo como entrada fatos diferentes da aplicação anterior, e com isso, realizaremos uma inferência completa sobre esta determinada interpretação, c.q.d.

## 2.8 Resumo

Neste capítulo, foram apresentados os principais conceitos e resultados necessários à definição de um procedimento de inferência. A partir das definições de base de regras, vetor de fatos, matrizes C e D, operadores matriciais conjuntivo "$\dot{\wedge}$" e disjuntivo "$\dot{\vee}$", passo de inferência e ciclo de inferência, mostrou-se que o procedimento de inferência apresentado implementa um mecanismo de inferência válido, dentro dos moldes da lógica proposicional.

No capítulo seguinte, serão apresentadas extensões deste procedimento de inferência, envolvendo a análise e a manipulação da base de regras, que poderão incrementar o desempenho do mesmo.

# 3. ANÁLISE E MANIPULAÇÃO DA BASE DE CONHECIMENTO

## 3.1 Introdução

A estrutura dos sistemas especialistas foi concebida de uma maneira tal que a separação entre o conhecimento e o mecanismo de manipulação de conhecimento é uma propriedade fundamental.

Em aplicações de tempo real, entretanto, a despeito de termos uma estrutura eficaz de manipulação do conhecimento (máquina de inferência) fica claro que é também necessário obter uma representação apropriada deste conhecimento, pois o mesmo deverá ser acessado e manipulado sob restrições temporais bem definidas. Caso o conhecimento não esteja organizado de maneira adequada, mesmo contando com estruturas ágeis na manipulação do conhecimento, corremos o risco de não conseguir realizar a inferência dentro do intervalo de tempo necessário, inviabilizando sua aplicação em muitos sistemas de controle.

Neste capítulo, apresentaremos dois métodos que buscam obter uma representação do conhecimento mais eficiente, principalmente para aplicações em sistemas de controle. Estes métodos fundamentam-se nas estruturas básicas de representação do conhecimento colocadas no capítulo 2, ou seja as matrizes operacionais C e D.

## 3.2 Análise das Bases de Regras

Quando falamos em base do conhecimento, incluimos tanto a base de regras como a base de fatos. Entretanto, para efeito de análise, consideraremos apenas a base de regras, que mostra-se ser a parte mais flexível para uma possível otimização.

No capítulo 2, mostramos que a base de regras pode ser totalmente descrita em termos das matrizes operacionais C e D. Deste modo, o que se deseja é realizar uma manipulação destas matrizes, de modo a obter um novo par de matrizes que, a despeito de contar com o mesmo conteúdo semântico das matrizes originais, possibilitem uma inferência mais eficiente, tanto em termos de ocupação de espaço como em tempo de execução do procedimento de inferência.

Um dos fatores que devem ser analisados dentro deste contexto é a questão do encadeamento de regras. Como se sabe, o encadeamento das regras é originado pelo fato de termos em um antecedente de uma regra, um consequente de outra regra. Com isso, a execução de um único passo de inferência não realizará a inferência completa, pois neste primeiro passo, será deduzida uma entidade, que seria necessária para disparar a segunda regra. Com isso, para se realizar a inferência completa (supondo que temos somente estas duas regras), seria necessário um outro passo de inferência, de modo que, a entidade inferida no primeiro passo possa no segundo passo validar o antecedente da segunda regra, fazendo com que seu consequente possa então ser deduzido. Como podemos ver, o encadeamento leva, em grandes bases de regras, a uma execução de um passo de inferência completo, às vezes por causa de "uma" regra que está encadeada. Supondo um número genérico de cadeias, podemos supor que seja necessária a realização de até m passos de inferência (m é o número de regras da base), desde que todas as regras da base estejam encadeadas. Isto significaria, em princípio, multiplicar por m o tempo de execução de um passo de inferência. Supondo que tenhamos uma base com um grande número de regras, o encadeamento poderá levar o tempo de execução do ciclo de inferência a dimensões não compatíveis com os requisitos de tempo real. Deste modo, a eliminação do encadeamento é uma das chaves para se obter uma melhora do desempenho. Dentro desta concepção propõe-se um método de expansão horizontal de uma base de regras (descrito na seção 3.3.1).

Outro fator que deve ser questionado dentro da meta de racionalizarmos o uso de uma base de regras, é a necessidade absoluta de que se preserve, durante o processo de inferência, todas as entidades citadas na base de regras original. Se quisermos nos ater à lógica clássica, ou lógica pura, este conceito faz-se necessário para que possamos manter a coerência de todos os teoremas sobre os quais se apoiam os processos de inferência. Entretanto, se estamos na realidade utilizando esta mesma lógica aplicada em sistemas de controle, onde a preocupação básica não é a manipulação da lógica matemática pura, mas sim o uso do conhecimento como maneira de se representar uma heurística, e que por meio desta se possa controlar uma atuação sobre um determinado processo, podemos nos desfazer de informações redundantes ou que não sejam determinantes na ação de controle. Com isso reduz-se a informação a ser processada, aumentando o desempenho computacional tanto a nível de tempo como de espaço.

Dentro deste espírito, propõe-se então um método de compressão de uma base de regras, onde informações desnecessárias serão desprezadas durante o processo de inferência. Este método será colocado na seção 3.3.2.

# 3.3 Manipulação da Base de Regras

Esta seção apresenta os métodos de expansão horizontal e de compressão de uma base de regras., Com isso espera-se obter uma base de regras onde o procedimento de inferência possa ser aplicado de forma a privilegiar o desempenho computacional.

## 3.3.1 Expansão Horizontal de uma Base de Regras

O método de expansão horizontal de um base de regras tem por meta a eliminação do encadeamento de uma base de regras, durante o procedimento de inferência, incluindo o encadeamento (que se faz necessário devido à propria estrutura do conhecimento envolvido) durante a fase de montagem da própria base de regras operacional. Note que esta montagem pode ser feita "off-line", ou seja, sem afetar os requisitos de tempo real.

A idéia básica é a de efetuar o encadeamento por meio da ampliação do conjunto de regras, onde as regras acrescentadas seriam na verdade as regras frutos dos encadeamentos das diversas outras regras, por silogismo. Deste modo, ao invés de ter que se efetuar um novo passo de inferência para determinar o encadeamento, ou seja a re-aplicação de todo o conjunto de regras, o que se faz é acrescentar uma única regra ao mesmo, com a qual se obtém o mesmo resultado que seria de se esperar após o encadeamento das diversas regras. Com isso, teremos uma base de regras horizontal, ou seja, onde a simples aplicação de um passo de inferência constitui o ciclo completo de inferência.

O método de expansão horizontal constitui então um procedimento pelo qual a partir das matrizes C e D originais, possa-se gerar automaticamente novas matrizes C' e D' implicando na necessidade de somente um passo para uma inferência completa.

Sabe-se que, se existe encadeamento, o mesmo é causado por uma entidade citada como consequente de alguma regra. O método consiste portanto, em se efetuar uma varredura no conjunto de regras, verificando em cada uma o seu consequente. Toma-se então esta entidade verificando para cada regra do conjunto-base (conjunto de regras original, antes de se iniciar a expansão) se este consequente é citado como antecedente. Em caso positivo, anexa-se ao conjunto de entidades do antecedente da regra verificada, os antecedentes da regra do conjunto-base, excetuando-se obviamente o consequente da regra verificada, e modifica-se o consequente da regra para o consequente da regra do conjunto-base. Esta nova regra é então acrescentada ao conjunto de regras. Esta varredura se faz inclusive sobre as regras acrescentadas.

A figura 3.1 mostra um exemplo de expansão horizontal

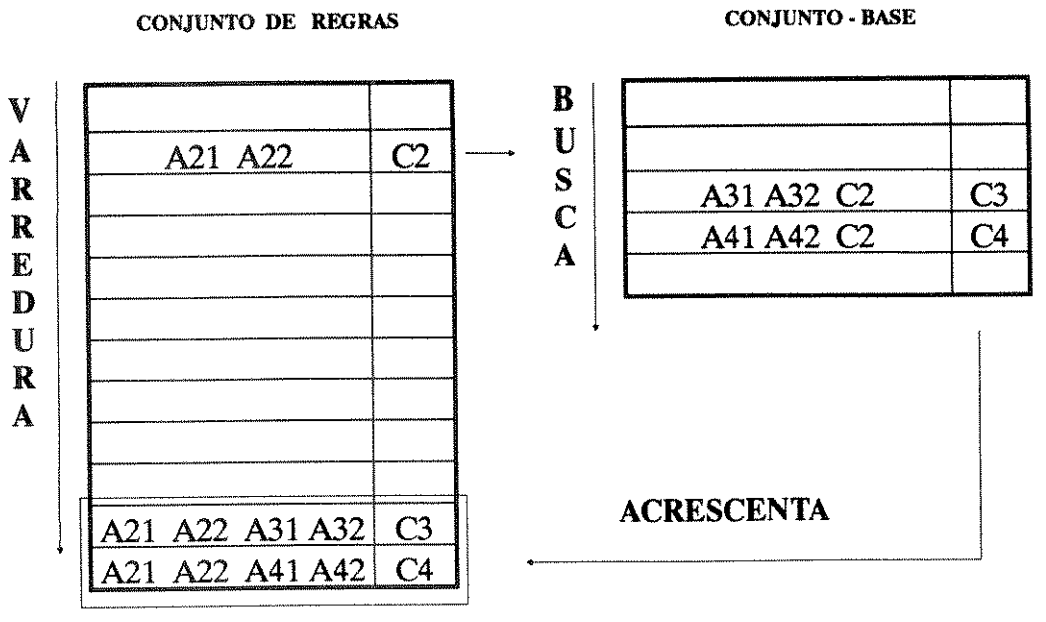


FIGURA 3.1 - Exemplo de Expansão Horizontal

A estrutura da matriz C é especialmente interessante para se realizar a expansão horizontal, visto que a procura de uma determinada entidade $e_j$ no antecedente da regra i, pode ser feita simplesmente verificando-se se $c_{ij}$ é igual a 1. Com isso, evita-se ficar percorrendo longas listas, durante a implementação.

## DEFINIÇÃO 3.1 : Base de Regras Horizontal

Define-se uma base de regras horizontal como uma base de regras em que o ciclo de inferência corresponde à aplicação de apenas um único passo de inferência.

## DEFINIÇÃO 3.2 : Procedimento de Expansão Horizontal

Seja $E = \{ e_i \}$ o conjunto de entidades do universo do discurso.
Seja BR uma base de regras.
Sejam C e D as matrizes operacionais que codificam a base BR.
Sejam $C_{i*}$ a i-ésima linha e $C_{*j}$ a j-ésima coluna da matriz C.
Sejam $D_{i*}$ a i-ésima linha e $D_{*j}$ a j-ésima coluna da matriz D.
Seja o " + " o operador vetorial de soma booleana.
Seja m o número de regras de BR, ou seja, o número de linhas da matriz C.

Seja m' o número de regras da base BR após (ou durante) o método da expansão horizontal.

Define-se o procedimento de expansão horizontal da base BR como sendo:

```
procedimento EXPANSÃO_HORIZONTAL
VAR
  i,j : índices;
INICIO
  m' := m;
  PARA i = 1 ATÉ m'
  INICIO
       PARA j = 1 ATÉ m
       INICIO
            Ache k tal que dki = 1;                          (3.1)
            SE cjk = 1                                       (3.2)
            INICIO
                 C (m'+1)* := Ci* + Cj* ;                    (3.3)
                 c (m'+1)k := 0;                             (3.4)
                 D*(m'+1) := D*j ;                           (3.5)
                 m' := m' + 1;
            FINAL
       FINAL
  FINAL
FINAL
```

TEOREMA 3.1 : Teorema da Expansão Horizontal

Seja BR uma base de regras, codificada em termos das matrizes C e D. Então, a base de regras BR' obtida pelo procedimento de expansão horizontal, é uma base de regras horizontal.

**PROVA :**

O procedimento de expansão horizontal analisa todos os consequentes das regras (3.1), verificando se os mesmos são antecedentes de outras regras (3.2). Se forem, ou seja, existe a possibilidade de um encadeamento, ele soma os antecedentes da regra analisada com os da regra do conjunto base (3.3). Em seguida ele retira o elemento consequente da regra analisada deste conjunto (3.4) e coloca o consequente da nova regra como sendo o consequente do conjunto-base (3.5). Com isso, esta nova regra tem o efeito semelhante ao de se efetuar dois passos de inferência, inferindo em um passo só o consequente da regra analisada e a do conjunto base. Assim, cada regra acrescentada no conjunto de regras elimina uma possível cadeia, de modo que após a varredura completa da base, nenhum encadeamento a mais é formado. Com isso temos uma base horizontal, c.q.d.

Observa-se que apesar de no procedimento de expansão horizontal falarmos em dois conjuntos de regras, o conjunto analisado e o conjunto base, na verdade os dois são um só. A diferença reside na parcela de regras que é utilizada na varredura. Com isso vemos que o procedimento é realmente um procedimento de expansão, pois transforma a dimensão m das matrizes (C(m

×n) e D(n ×m)) em m' ≥ m. Deste modo, apesar de se ganhar com o fim do encadeamento no procedimento de inferência, exigindo apenas um passo de inferência, aumenta-se o tamanho físico da base de regras. Logicamente existe um compromisso entre o benefício da redução do tempo de inferência e o aumento do espaço de armazenamento da base de regras, mas normalmente, em caso de bases com poucos encadeamentos, a expansão horizontal se mostra altamente eficiente, induzindo uma melhora sensível no desempenho do sistema. Se aliarmos então a expansão horizontal à compressão da base de regras (a ser abordada na próxima seção), veremos que então nem mesmo haverá a necessidade de se aumentar a base de regras, tornando o procedimento de inferência incomparavelmente mais rápido e eficiente do que os procedimentos anteriores.

### 3.3.2 Compressão de uma Base de Regras

Na seção anterior, verificamos o método de expansão horizontal de uma base de regras. Por meio desta expansão horizontal, consegue-se transformar uma base de regras genérica em uma base de regras horizontal, necessitando para a inferência somente um passo, não havendo encadeamento.

Em sistemas de controle, normalmente temos sinais que são extraídos do processo considerado, que são convertidos então em fatos, por meio de, por exemplo, pré-processamento. Estes fatos são então enviados à máquina de inferência, para após a inferência, gerar os fatos (decisões) os quais após o pós-processamento, tornar-se-ão os sinais de controle.

Durante o processo de inferência, são citadas várias entidades intermediárias, de modo que através do encadeamento entre elas, se chegue à inferência dos fatos-decisão. Estas entidades intermediárias são importantes, pois é por meio delas que o ser humano expressa as regras de produção que traduzem a heurística, ou conhecimento específico, que se deseja representar. Entretanto, apesar de importantes na fase de aquisição do conhecimento, estas entidades tornam-se totalmente dispensáveis no procedimento de inferência em tempo real, visto que nestes tipos de sistema, não é necessária a explicação do porquê de cada dedução, mas se leva em consideração apenas a efetividade da decisão e o tempo em que ela foi tomada. Com isso, nestes tipos de sistema, desde que eliminemos os encadeamentos, podemos eliminar estas entidades intermediárias. Supondo que a base de regras tenha sido expandida horizontalmente, não haverá encadeamento, podendo-se portanto dispensar tais entidades intermediárias.

Por outro lado, não existe necessidade de colocarmos na entrada do sistema, nem mesmo de processarmos, as entidades que sabidamente só irão assumir valor verdade após o procedimento de inferência. Com isso, podemos dividir nosso vetor de fatos em dois sub-vetores: sub-vetor de entrada e sub-vetor de saída. Supondo o procedimento de inferência do capítulo 2, onde temos um vetor de fatos de entrada e um vetor de fatos inferidos, vemos que o sub-vetor de saída do vetor de fatos de entrada não é necessário. Do mesmo modo, o sub-vetor de entrada do vetor de fatos inferidos deve ser idêntico ao do vetor de fatos de entrada, não sendo portanto importante que guardemos uma

informação replicada, podendo também ser desprezado. Por sua vez, os elementos das matrizes C e D que dizem respeito a toda esta informação inútil, também podem ser desprezados.

Com isso, vemos que podemos desprezar aquelas informações que não são interessantes para um sistema voltado para aplicações em tempo real, onde o número de informações a ser processada pode ser muito grande e é mais importante a informação obtida pelo processo de inferência do que a maneira como esta informação foi obtida.

Ao processo de identificação e eliminação desta informação desnecessária chamouse de compressão de uma base de regras. Para realizarmos a compressão é necessário antes fazermos algumas distinções entre as entidades.

Durante o procedimento de expansão horizontal, é possível qualificarmos alguns atributos às entidades manipuladas. As entidades que pertencem somente aos antecedentes das regras, são então chamadas de entidades de entrada. As entidades que são citadas tanto nos antecedentes das regras como nos consequentes são chamadas de entidades intermediárias. Aquelas que são citadas somente nos consequentes das regras são chamadas de entidades de saída. De posse destes atributos, podemos definir o procedimento de compressão.

Após a expansão horizontal, toma-se o conjunto de entidades intermediárias e elimina-se as mesmas do vetor de fatos de entrada, do vetor de fatos inferidos. Do mesmo modo, retira-se da matriz C as colunas referentes a estas entidades. Da matriz D, elimina-se então as linhas correspondentes às mesmas. Retira-se então do vetor de fatos de entrada, as entidades de saída. Do mesmo modo, retira-se as colunas da matriz C relativas a estas entidades. Retira-se então do vetor de fatos inferidos as entidades de entrada, e da matriz D as linhas relativas a estas entidades. Com isso, teremos então a base comprimida.

É necessário no entanto, modificarmos ligeiramente o passo de inferência definido anteriormente. Sendo n1 o número de entidades de entrada e n2 o número de entidades de saída e uma vez que não existe agora mais nenhum vínculo entre o vetor de fatos de entrada e o vetor de fatos inferidos, (seus elementos representam entidades diferentes), não é mais necessário que se faça a soma boolana entre o vetor de fatos de entrada e o resultado dos operadores matriciais (o que nem seria correto). Com isso, o passo de inferência fica simplesmente:

$$xi = D \dot{\wedge} (\overline{C} \dot{\vee} xe) \qquad (3.6)$$

DEFINIÇÃO 3.3 : Compressão de Uma Base de Regras

Seja BR uma base de regras.

Seja BR' uma base de regras horizontal, fruto da expansão horizontal de BR.

Sejam C e D as matrizes operacionais que codificam a base BR'.

Seja E o conjunto de entidades do universo de discurso.

Seja $A_j$ o conjunto de entidades pertencentes ao antecedente da regra $R_j$.

Seja $A$ o conjunto formado pela união de todos os conjuntos $A_j$, ou seja, os antecedentes de todas as regras.

Seja $Q_j$ o conjunto de entidades pertencentes ao consequente da regra $R_j$

Seja $Q$ o conjunto formado pela união de todos os conjuntos $Q_j$, ou seja, os consequentes de todas as regras.

Seja N o conjunto de entidades de entrada, ou seja, o conjunto das entidades citadas somente nos antecedentes das regras de BR'.

$$N = \left\{ n_i \mid n_i \in A , n_i \notin Q , 1 \leq i \leq \text{Card}\,(A - (A \cap Q)) \right\} \qquad (3.7)$$

Seja I o conjunto de entidades intermediárias, ou seja, o conjunto das entidades citadas nos antecedentes e nos consequentes das regras de BR'.

$$I = \left\{ i_k \mid i_k \in A , i_k \in Q , 1 \leq k \leq \text{Card}\,(A \cap Q) \right\} \qquad (3.8)$$

Seja S o conjunto de entidades de saída, ou seja, o conjunto das entidades citadas somente nos consequentes das regras de BR'.

$$S = \left\{ s_l \mid s_l \in Q , s_l \notin A , 1 \leq l \leq \text{Card}\,(Q - (A \cap Q)) \right\} \qquad (3.9)$$

Naturalmente tem-se que $E = N \cup I \cup S$.

Sejam xe o vetor de fatos de entrada e xi o vetor de fatos inferidos.

Define-se então o procedimento de compressão de uma base de regras como:

procedimento COMPRESSÃO

INICIO
Retirar todo $e_i \in I$ de xe e xi. (3.10)
Retirar de C todas as linhas $C_{i*}$ tal que $\exists j$ $(e_j \in I \wedge c_{ij} = 1)$. (3.11)
Retirar de D todas as colunas correspondentes $D_{*i}$ . (3.12)
Retirar de C todas as colunas $C_{*j}$ tal que $e_j \in I$. (3.13)
Retirar de D todas as colunas $D_{*j}$ tal que $\exists i( e_i \in I \wedge d_{ij} = 1)$. (3.14)
Retirar de C todas as linhas correspondentes $C_{j*}$ . (3.15)
Retirar de D todas as linhas $D_{i*}$ tal que $e_i \in I$ (3.16)
Retirar todo $ek \in S$ de xe. (3.17)
Retirar todo $el \in N$ de xi. (3.18)
Retirar de C todas as colunas $C_{*j}$ tal que $e_j \in S$ (3.19)
Retirar de D todas as linhas $D_{i*}$ tal que $e_i \in N$. (3.20)
FINAL

O procedimento de retirar um elemento de um vetor de fatos corresponde à supressão do mesmo do vetor, com o consequente deslocamento da parte final do vetor preenchendo a sua posição, ou seja:

Sendo x um vetor de fatos:

$$x = \left[ \; x_1, x_2, \ldots, x_{p-1}, x_p, x_{p+1}, \ldots, x_n \; \right]$$

a retirada do elemento $e_p$ ,representado por $x_p$, do vetor x corresponde a criar um vetor

$$x' = \left[ \; x_1, x_2, \ldots, x_{p-1}, x_{p+1}, \ldots, x_n \; \right]$$

onde a dimensão do vetor x' é igual a dimensão de x decrementada de um.

Do mesmo modo, a retirada de uma linha ou coluna de uma matriz corresponde à supressão da linha/coluna da mesma, com o consequente deslocamento das outras linhas/colunas preenchendo sua posição.

Observa-se que, com este procedimento, modifica-se o índice correspondente às entidades utilizadas e no caso das mesmas serem referenciadas em alguma tabela, a mesma deve ser atualizada de maneira semelhante.

TEOREMA 3.2 : Teorema da Compressão

Seja BR uma base de regras.
Seja BR' a base de regras BR expandida horizontalmente.
Sejam C e D as matrizes operacionais que codificam BR'.

Sejam N o conjunto de entidades de entrada, I o conjunto de entidades intermediárias e S o conjunto de entidades de saída.

Seja xe um vetor de fatos de entrada.

Seja xi um vetor de fatos inferidos.

Seja xe' um vetor de fatos modificado para o conjunto N

Seja xi' um vetor de fatos inferidos modificado para o conjunto S.

Seja Comp(C,D,xe,xf,C',D',xe',xf') o procedimento de compressão de uma base de regras que transforma uma base BR' em BR", conforma a definição 3.3.

Sejam C' e D' as matrizes operacionais que codificam a base BR" comprimida.

Então, o valor verdade assumido pelos elementos do vetor xi', pelo procedimento de inferência modificado, utilizando as matrizes operacionais comprimidas C' e D'

$$xi' = D' \dot{\wedge} ( \overline{C'} \dot{\vee} xe')$$

é idêntico ao que seria assumido pelos elementos de xi correspondentes às entidades de xi', utilizando-se o procedimento convencional e as matrizes C e D, para um vetor de entrada xe onde as entidades correspondentes a xe' tenham valor verdade semelhante aos de xe' e as demais tenham valores iguais a zero.

**PROVA :**

Analisando o procedimento de compressão, vemos que em (3.10) retira-se do conjunto de entidades operacionais, as entidades intermediárias. Em (3.11) e (3.12) retiram-se todas as regras que tenham em seu antecedente uma entidade intermediárial. Em (3.13), retiram-se do conjunto de possíveis antecedentes as entidades intermediárias, de modo a tornar as linhas da matriz C uma representação consistente para a comparação com o vetor de fatos de entrada modificado em (3.10). Em (3.14) e (3.15) retiram-se as regras que tenham em seu consequente uma entidade intermediária. Em (3.16) retiram-se do conjunto de possíveis consequentes as entidades intermediárias, de modo a tornar as colunas da matriz D uma representação consistente para o vetor de fatos inferidos, que foi modificado em (3.10).

Nestes primeiros passos, vemos que os mesmos são equivalente à retirar as entidades intermediárias da base de regras e do vetor de fatos.

Isto seria equivalente a definir um novo sistema, onde o universo de discurso se reduziria a (N $\cup$ S) e as regras seriam as mesmas de BR', excetuando-se aquelas que citem alguma entidade intermediária. Esta seria uma compressão intermediária, e conforma a proposição do teorema, onde somente as entidades de N podem assumir valores diferentes de 0, (visto que xe' representa exclusivamente as entidades de N), temos imediatamente que, como somente as regras que citem as entidades intermediárias foram removidas, os valores verdade inferidos por C' e D' e o procedimento modificado, seriam semelhantes aos inferidos por C e D e o procedimento convencional, conforme diz o teorema. Em (3.17) retiram-se do vetor de fatos de entrada os elementos de saída e em (3.18) retiram-se do vetor de fatos inferidos os elementos de

entrada. Em (3.19) ajustam-se as colunas de C para a nova representação colocada em (3.17) e em (3.20) ajustam-se as linhas de D para a nova representação colocada em (3.18).

De novo, de acordo com a proposição do teorema, os valores de xe devem ser tais que seus valores verdade sejam equivalentes aos de xe', tendo os demais em zero. Deste modo, se os demais são zero, não irão interferir no processo de inferência, para o caso convencional, fazendo com que os resultados sejam semelhantes. Do mesmo modo, a retirada das entidades de entrada do vetor de fatos inferidos, não influi em nada, visto que não são citadas no vetor modificado xf'. Deste modo, os valores são semelhantes, conforme diz o teorema, e o mesmo é então provado, c.q.d.

## 3.4 Exemplo de Aplicação dos Procedimentos

Nesta seção é apresentado um exemplo simples colocado de modo a ilustrar a aplicação dos procedimentos desenvolvidos até agora.

Considere a seguinte base de regras:

SE $e_1$ E $e_2$ ENTÃO $e_3$
SE $e_3$ E $e_4$ ENTÃO $e_5$
SE $e_1$ E $e_6$ ENTÃO $e_5$
SE $e_2$ E $e_6$ ENTÃO $e_7$
SE $e_5$ E $e_6$ ENTÃO $e_7$
SE $e_2$ E $e_4$ ENTÃO $e_8$

Define-se então a seguinte codificação:

| SÍMBOLO | ÍNDICE | ANTEC. | CONS. | TIPO |
|---|---|---|---|---|
| $e_1$ | 1 | SIM | NÃO | N |
| $e_2$ | 2 | SIM | NÃO | N |
| $e_3$ | 3 | SIM | SIM | I |
| $e_4$ | 4 | SIM | NÃO | N |
| $e_5$ | 5 | SIM | SIM | I |
| $e_6$ | 6 | SIM | NÃO | N |
| $e_7$ | 7 | NÃO | SIM | S |
| $e_8$ | 8 | NÃO | SIM | S |

A partir desta codificação, montam-se as matrizes C e D, a seguir:

$$C = \begin{bmatrix} 1\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0 \\ 0\,1\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,1\,0\,0 \\ 0\,1\,0\,1\,0\,0\,0\,0 \end{bmatrix} \quad D = \begin{bmatrix} 0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,1\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,1 \end{bmatrix}$$

Após executar o procedimento de Expansão Horizontal, obtém-se as matrizes C' e D':

$$C' = \begin{bmatrix} 1\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0 \\ 0\,1\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,1\,0\,0 \\ 0\,1\,0\,1\,0\,0\,0\,0 \\ 1\,1\,0\,1\,0\,0\,0\,0 \\ 0\,0\,1\,1\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,1\,0\,0 \\ 1\,1\,0\,1\,0\,1\,0\,0 \end{bmatrix} \quad D' = \begin{bmatrix} 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0 \end{bmatrix}$$

Executando-se então o procedimento de Compressão, têm-se uma nova tabela de símbolos, de modo a representar os novos vetores base de fatos de entrada e base de fatos inferidos:

| VETOR DE FATOS DE ENTRADA | |
|---|---|
| SÍMBOLO | ÍNDICE |
| $e_1$ | 1 |
| $e_2$ | 2 |
| $e_4$ | 3 |
| $e_6$ | 4 |

| VETOR DE FATOS DE SAÍDA | |
|---|---|
| SÍMBOLO | ÍNDICE |
| $e_7$ | 1 |
| $e_8$ | 2 |

e as correspondentes matrizes comprimidas C" e D":

$$C'' = \begin{bmatrix} 0\,1\,0\,1 \\ 0\,1\,1\,0 \\ 1\,0\,0\,1 \\ 1\,1\,1\,1 \end{bmatrix} \quad D'' = \begin{bmatrix} 1\,0\,1\,1 \\ 0\,1\,0\,0 \end{bmatrix}$$

Suponha que os sinais coletados pelo processo foram pré-processados, resultando nos seguintes dados:

| DADOS PRÉ_PROCESSADOS | |
|---|---|
| PROPOSIÇÃO | VALOR VERDADE |
| $e_1$ | FALSE |
| $e_2$ | TRUE |
| $e_4$ | FALSE |
| $e_6$ | TRUE |

Então o vetor de fatos de entrada será:

$$xi' = [\,0\,1\,0\,1\,]$$

O resultado fornecido pelo procedimento de inferência será:

$$xo' = [\,1\,0\,]$$

Assim, a ação de controle correspondente a $e_7$ será executada pelo pós-processamento. A decisão correspondente a $e_8$ não será executada.

## 3.5 Resumo

Neste capítulo apresentou-se uma breve análise sobre a importância da estrutura da base de regras no desempenho do procedimento de inferência, e mostrou-se dois métodos pelos quais pode-se transformar uma base genérica em uma base estruturada de maneira a incrementar este desempenho, que são o método da expansão horizontal e da compressão.

No capítulo seguinte, será colocada a teoria que existe por trás da manipulação de elementos de conjuntos nebulosos, gerando a base da lógica nebulosa. Serão definidos estes conceitos básicos, seguidos de especificações de controle nebuloso e inferência nebulosa.

# 4. TEORIA DOS CONJUNTOS NEBULOSOS E LÓGICA NEBULOSA

## 4.1 Introdução

Neste capítulo faz-se um rápido sumário sobre a teoria de conjuntos nebulosos (ou "fuzzy") e lógica nebulosa, de modo a proporcionar um substrato teórico às colocações que virão no capítulo 5, onde o procedimento de inferência via matrizes C-D é extendido para aplicações em lógica nebulosa.

## 4.2 Elementos de Conjuntos Nebulosos - Regra Nebulosa

A teoria dos conjuntos nebulosos foi formulada inicialmente por Zadeh em [41], como uma opção para a modelagem de sistemas complexos, em que um certo grau de incerteza nos parâmetros do modelo não influiria de maneira significativa no controle do sistema. Em seguida, diversos autores desenvolveram trabalhos sobre as idéias básicas de Zadeh, utilizando-as, dentre outras áreas, em sistemas de Inteligência Artificial [8,20,22,24,25,27] e em sistemas de controle [17,18,23,26,31,34,40]. Particularmente, em [17,18] têm-se um resumo interessante sobre as aplicações em sistemas de controle genéricos e em [26,27] uma comparação importante sobre os diversos tipos de operadores que podem ser utilizados no processo de inferência de regras nebulosas.

Para a definição de um conjunto nebuloso, leva-se em conta que a seus elementos não é dada uma pertinência absoluta. Em princípio, qualquer elemento de um universo de discurso pode pertencer a um conjunto nebuloso. A pertinência de tal elemento é dada por uma função de pertinência, ou função característica do conjunto, que associa a cada elemento do universo de discurso, um grau de pertinência referente a quanto o elemento pertence ao conjunto. Esta função é definida de forma a ter como domínio o universo de discurso e contradomínio o intervalo [0,1]. Deste modo, a interpretação dada a um conjunto nebuloso deixa de ser "se um elemento pertence ao conjunto", mas "quanto ele pertence ao conjunto", pois ao contrário dos conjuntos convencionais, onde existe uma pertinência absoluta (ou um elemento pertence ou não pertence a um conjunto convencional), os conjuntos nebulosos são caracterizados pela existência de uma gradação da pertinência de seus elementos.

As definições básicas sobre conjuntos nebulosos podem ser encontradas em [41] ou [17], sendo descritas a seguir:

## DEFINIÇÃO 4.1: Conjunto Nebuloso

Seja U uma coleção de objetos denotados genericamente por {u}, que podem ser discretos ou contínuos. U é chamado de universo de discurso sendo "u" um elemento genérico de U.

Deste modo um conjunto nebuloso N em um universo de discurso U é caracterizado pela função de pertinência $\mu_N$ que assume valores no intervalo [0,1]. Ou seja:

$$\mu_N : U \in [0,1]$$

Um conjunto nebuloso pode ser visto como uma generalização de um conjunto ordinário cuja função de pertinência assume valores em {0,1}. Deste modo um conjunto nebuloso N em U pode ser representado como um conjunto de pares ordenados de elementos u e seu respectivo grau de pertinência:

$$N = \{(u, \mu_N (u)) \mid u \in U \}$$

Quando U é contínuo, um conjunto nebuloso N pode ser escrito de modo conciso como:

$$N = \int_u \mu_N (u)/u$$

Quando U é discreto, o conjunto nebuloso N pode ser representado por:

$$N = \sum_1^n \mu_N (u)/u$$

onde os símbolos de integral e somatória se referem ao seu sentido dentro da teoria de conjuntos, ou seja, é a união de termos.

## DEFINIÇÃO 4.2 : Conjunto Suporte, Ponto de Limiar e Conjunto Unitário

Define-se conjunto suporte como o conjunto de todos os pontos u em U tal que $\mu_N(u) = 0$. Ou seja, é o sub-conjunto do universo de discurso que tem o grau de pertinência de seus elementos diferente de zero. Particularmente, o ponto onde $\mu_N = 0.5$ é chamado de ponto de limiar.

Um conjunto nebuloso que tem como suporte um único elemento, e além disso, para este elemento $\mu_N = 1.0$, é chamado de conjunto unitário

Sejam A e B dois conjuntos nebulosos em U com funções de pertinência $\mu_A$ e $\mu_B$ respectivamente. Operações sobre conjuntos tais como união,

intersecção e complemento são definidas para conjuntos nebulosos através de suas funções de pertinência.

### DEFINIÇÃO 4.3 União

A função de pertinência $\mu_C$ de $C = A \cup B$ é definida ponto a ponto para todo u e U:

$$\mu_C = \max \{ \mu_A(u) , \mu_B(u) \}$$

### DEFINIÇÃO 4.4 Intersecção

A função de pertinência $\mu_C$ de $C = A \cap B$ é definida ponto a ponto para todo $u \in U$:

$$\mu_C = \min \{ \mu_A(u) , \mu_B(u) \}$$

### DEFINIÇÃO 4.5 Complemento

A função de pertinência $\mu_C$ de $C = \overline{A}$ é definida ponto a ponto para todo $u \in U$:

$$\mu_C = 1 - \mu_A$$

### DEFINIÇÃO 4.6 Norma Triangular e Co-Norma Triangular

Um operador T é chamado de norma triangular se e somente se satisfaz as seguintes condições:

(i) $T(0,0) = 0$
(ii) $T(x,1) = x$
(iii) $T(x,y) \leq T(u,v)$, sempre que $x \leq u$ e $y \leq v$
(iv) $T(x,y) = T(y,x)$
(v) $T(T(x,y),z) = T(x,T(y,z)) = T(x,y,z)$

Um operador S é chamado de co-norma triangular se e somente se satisfaz as condições (iii),(iv) e (v) anteriores e além destas as seguintes, que substituem as condições (i) e (ii) respectivamente:

(i') $S(1,1) = 1$
(ii') $S(x,0) = x$

As normas triangulares e co-normas triangulares podem ser associadas mutuamente. Dada uma norma triangular T:

$$S(x,y) = 1 - T(1-x,1-y)$$

Como exemplo de normas triangulares temos, por exemplo:

intersecção $x \wedge y = \min(x, y)$
produto algébrico $x \times y = xy$
produto limitado $x \otimes y = \max(0, x+y-1)$
produto drástico $x \dot{\cap} y = \begin{cases} x \text{ se } y = 1 \\ y \text{ se } x = 1 \\ 0 \text{ se } x, y < 1 \end{cases}$

As respectivas co-normas triangulares são

união $x \vee y = \max(x, y)$
soma algébrica $x \pm y = x + y - xy$
soma limitada $x \oplus y = \min(1, x+y)$
soma drástica $x \dot{\cup} y = \begin{cases} x \text{ se } y = 0 \\ y \text{ se } x = 0 \\ 1 \text{ se } x, y > 0 \end{cases}$

DEFINIÇÃO 4.7 Produto Cartesiano

Sejam $A_1 \ldots A_n$ conjuntos nebulosos definidos nos universos de discurso $U_1 \ldots U_n$ respectivamente. O produto cartesiano de $A_1 \ldots A_n$ é um conjunto nebuloso que tem por universo de discurso $U_1 \times \ldots \times U_n$. Sendo $S = A_1 \times \ldots \times A_n$ tem-se que:

$$\mu_S(u_1, u_2, \ldots, u_n) = \text{norm\_trian.}\left(\mu_{A1}(u_1), \mu_{A2}(u_2), \ldots, \mu_{An}(u_n)\right)$$

onde norm_trian. diz respeito a uma norma triangular.

DEFINIÇÃO 4.8 Relação Nebulosa

Uma relação nebulosa n-ária em um universo de discurso $U_1 \times \ldots \times U_n$ é dada pelo produto cartesiano n-ário:

$$R = \left\{ (u_1, \ldots, u_n), \mu(u_1, \ldots, u_n) \mid (u_1, \ldots, u_n) \in U_1 \times \ldots \times U_n \right\}$$

DEFINIÇÃO 4.9 Composição

Sejam R e S duas relações nebulosas em $U \times V$ e $V \times W$ respectivamente. A composição de R e S é definida como uma relação nebulosa representada por $R \bullet S$, definida como:

$$R \bullet S = \left\{ [(u, w), \sup (\mu_R(u,v) * \mu_S(v,w)], u \in U, v \in V, w \in W \right\}$$

onde * corresponde a uma norma triangular.

Os conjuntos nebulosos são importantes pois conseguem representar conceitos, entendendo-se por conceito um conhecimento de certo modo impreciso ou difuso, mas que pode ser exprimido e manipulado de modo a gerar decisões.

A linguagem humana serve-se frequentemente do uso de conceitos para a descrição de conhecimentos. Conceitos como alto, baixo, frio, quente, são mais do que usuais na comunicação entre seres humanos sendo utilizados durante o raciocínio para a tomada de decisões. Por exemplo, considere o ajuste da temperatura de um chuveiro elétrico:

- Quando a água está muito fria, fecha-se um pouco a torneira, de modo que a vazão de água diminua um pouco e com isso a água fique um pouco mais quente.

Esta descrição de um raciocínio humano está recheada de conceitos, tais como muito fria, fechar um pouco, diminuir um pouco, um pouco mais quente, que apesar de não estarem quantificados, em termos da temperatura absoluta da água, o número de voltas da torneira, vazão em litros por segundo e taxa de variação da temperatura da água, conseguem expressar um conhecimento que é manipulado pelo cérebro humano, gerando decisões que podem levar ao controle da temperatura da água.

Este tipo de conhecimento pode ser descrito por intermédio de variáveis linguísticas. Uma variável linguística é definida como uma variável que assume como valores, conceitos, ao invés de números. Deste modo, uma variável linguística temperatura, pode assumir como valores, por exemplo, os conceitos baixa, ou alta, assim como uma variável linguística água pode assumir os valores quente ou fria. Como um conceito pode ser representado por meio de um conjunto nebuloso, pode-se descrever uma variável linguística como uma variável que tenha por atributo conjuntos nebulosos, onde o valor desta variável é associado ao conceito representado pelo conjunto nebuloso dado como atributo. Do mesmo modo, uma dada grandeza física pode estar relacionada a vários conceitos simultaneamente. Por exemplo, suponha-se uma variável linguística temperatura. O espectro de valores associados a temperatura pode ser dividido em diversos conceitos, ou partições, dentre elas baixa,média, alta, por exemplo. Nestas, os conceitos estão intercalados. Ou seja, para uma dada temperatura física, a mesma pode ser encarada como baixa, média e alta ao mesmo tempo, somente com diferentes graus de pertinência aos três conjuntos. Poderíamos ter, por exemplo, um grau de pertinência de 0.9 para o conjunto alta, um grau de 0.7 para média e 0.2 para baixa.

Os conceitos (ou partições nebulosas) podem ser associados por meio de conectivos lógicos, tais como a conjunção ("$\wedge$", "and", "e"), a disjunção ("$\vee$", "or" ,"ou") e a negação ("$\overline{x}$", "$\neg x$", "not(x)", "x negado", "não x"). Estes conectivos representam, em termos de lógica nebulosa, as idéias de intersecção, união e

complemento, respectivamente, de maneira semelhante a da da teoria de conjuntos.

A interpretação de tais conectivos se dará portanto conforme as definições 4.3, 4.4 e 4.5. Associações do tipo:

- muito quente ou muito frio
- médio e alto (no sentido de alto mas não tão alto)
- (não muito pequeno) e (não muito grande)

são perfeitamente válidas, com uma formulação bem definida utilizando-se as definições de intersecção, união e complemento para os conjuntos nebulosos.

Utilizando-se a idéia de variável linguística, pode-se definir também as proposições nebulosas. Uma proposição nebulosa é simplesmente a atribuição de um conceito, representado por um conjunto nebuloso, e do mesmo modo um conjunto nebuloso formado pela associação de vários conjuntos nebulosos por meio dos conectivos lógicos nebulosos, a uma variável linguística.

Assim, construções do tipo:

- temperatura é alta
- água é quente
- tamanho é ((não muito pequeno e (não muito grande))

ou do tipo:

- x é A
- y é B
- z é $(A \vee (\neg B)) \wedge C$

são chamadas de proposições nebulosas.

DEFINIÇÃO 4.10 Proposição Nebulosa

Define-se uma proposição nebulosa como a associação de um conjunto nebuloso, representando um conceito, a uma variável linguística.

Uma proposição nebulosa pode ser representada genericamente por uma construção do tipo (x é A) onde x é uma variável linguística e A é um conjunto nebuloso.

Do mesmo modo, pode ser representado também pelo par (x,A).

DEFINIÇÃO 4.11 Conjunção de Proposições Nebulosas

A conjunção de n proposições nebulosas $(x_1$ é $A_1)$,...,$(x_n$ é $A_n)$, representada genericamente por

$$(x_1 \text{ é } A_1) \text{ E } \ldots \text{ E } (x_n \text{ é } A_n)$$

pode ser unificada em uma proposição nebulosa n-ária

$$(x_1,...,x_n) \text{ é } A$$

onde $A$ é o produto cartesiano $A_1 \times ... \times A_n$

DEFINIÇÃO 4.12 Regra Nebulosa

Define-se uma regra nebulosa como uma regra de produção que utiliza proposições nebulosas do tipo ($u_i$ é $A_i$), onde $u_i$ é uma variável linguística e $A_i$ é um conjunto nebuloso representando um conceito. A regra é formada por um antecedente, representando uma condição, estruturado em termos de uma associação conjuntiva de proposições nebulosas e um consequente, uma outra proposição nebulosa. A estrutura básica de uma regra nebulosa é a seguinte:

$$\text{SE } (u_1 \text{ é } A_1) \text{ E } (u_2 \text{ é } A_2) \text{ E } ... \text{ E } (u_n \text{ é } A_n) \text{ ENTÃO } (y \text{ é } C)$$

Como exemplo, poderíamos ter (sendo x uma variável linguística de temperatura ,y uma variável linguística de pressão, e z uma variável linguística de injeção de calor) regras do seguinte tipo:

SE (x é baixa) E (y é alta) ENTÃO (z é média)

significando no caso que para uma temperatura baixa e uma pressão alta, deve-se ter uma injeção de calor média. O antecedente da regra pode ser considerado como uma proposição nebulosa, tendo uma variável linguística binária (x,y) cujo valor é o produto cartesiano baixa × alta, definido sobre o universo de discurso U × V, se baixa é definida em U e alta é definida em V. Ou seja, a regra é semelhante a:

SE [(x,y) é (baixa × alta)] ENTÃO (z é média)

(OBS: A definição de o que é alta, média ou baixa, é dada pelas funções de pertinência de cada conjunto, que por sua vez, é dada para cada tipo de variável linguística. Assim, o conjunto alta para temperaturas, é totalmente independente do conjunto alta para pressão e o mesmo para injeção de calor.)

Genericamente podemos tratar de uma regra de produção nebulosa do seguinte tipo:

$$\text{SE } (x_1 \text{ é } A_1) \text{ E } (x_2 \text{ é } A_2) \text{ E } ... \text{ E } (x_n \text{ é } A_n) \text{ ENTÃO } (y \text{ é } C)$$

Pela definição 4.11, o antecedente da regra pode ser descrito por uma proposição nebulosa n-ária, o que reduz a regra a seguinte forma:

$$\text{SE } [(x_1, x_2, ..., x_n) \text{ é } A] \text{ ENTÃO } (y \text{ é } C)$$

onde $A$ é dado por $A = A_1 \times A_2 \times ... \times A_n$

DEFINIÇÃO 4.13 Implicação Nebulosa

O conectivo ENTÃO utilizado na descrição de regras nebulosas corresponde ao operador nebuloso de implicação. Uma das maneiras de se definir a implicação nebulosa é por meio de uma relação nebulosa. Deste modo, uma regra do tipo

SE (x é A) ENTÃO (y é B)

correspondendo a (x é A) → (y é B) pode ser unificada em uma proposição nebulosa binária:

(x,y) é R

onde R é a relação binária (no caso) dada por $R = A \times B$.
Assim, uma regra nebulosa genérica corresponde à seguinte fórmula:

$$(x_1, x_2, ..., x_n) \text{ é } A \rightarrow y \text{ é } C$$

que pode ser reduzida a

$$(x_1, x_2, ..., x_n, y) \text{ é } (A \times C)$$

ou, supondo $R = (A \times C)$

$$(x_1, x_2, ..., x_n, y) \text{ é } R$$

DEFINIÇÃO 4.14 Modus Ponens

Define-se por Modus Ponens o seguinte procedimento de inferência em lógica proposicional.
Se A e A→B são proposições válidas, então deduz-se B:

$$\frac{A,\ A \rightarrow B}{B}$$

Diz-se então que B é fruto da composição de A e A→B, onde o operador composição é simplesmente o operador lógico booleano "and". Deste modo

$$B = A \bullet A \rightarrow B = A \wedge A \rightarrow B$$

DEFINIÇÃO 4.15 Modus Ponens Generalizado

O procedimento de Modus Ponens pode ser generalizado para incluir também a lógica nebulosa. Para tal, se serve de proposições nebulosas, da seguinte forma:

$$\frac{(x \text{ é } A'),\ (x \text{ é } A) \rightarrow (y \text{ é } B)}{(y \text{ é } B')}$$

Que significa que se (x é A') é uma proposição nebulosa e que (x é A) → (y é B) é uma proposição nebulosa (equivalente a (x,y) é (A × B)), deduz-se a proposição nebulosa (y é B'), onde B' é dado pela composição nebulosa:

$$B' = A' \bullet (A \times B)$$

Este mecanismo de inferência vale também para proposições nebulosas n-árias:

$$\frac{[(x_1, x_2, ..., x_n) \text{ é } R'],\ [(x_1, x_2, ..., x_n, y) \text{ é } R]}{y \text{ é } R''}$$

onde R'' = R' • R ("•" é o operador de composição conforme a definição 4.9).

Pode-se portanto efetuar inferências na lógica nebulosa a partir de uma extensão do mecanismo de inferência da lógica proposicional. A diferença básica, comparando-se com a lógica proposicional, é que os operadores nebulosos (conjunção de proposições nebulosas, composição nebulosa, implicação nebulosa) podem ser definidos de várias maneiras, o que certamente levará a diferentes resultados. A escolha dos operadores pode estar relacionada com algum tipo de distribuição probabilística que possa ser associada aos conjuntos nebulosos. Entretanto nem sempre será possível uma associação explícita deste tipo.

## 4.3 Sistema e Controle Nebuloso

A teoria de conjuntos nebulosos e lógica nebulosa pode ser utilizada para a descrição de modelos de sistemas físicos, de modo semelhante ao que foi feito para a lógica proposicional. Por meio de uma série de regras nebulosas, obtém-se um modelo do sistema físico que se deseja controlar.

No caso de controladores, a partir de sensores, coletam-se informações sobre o estado do sistema. Estas informações são então transformadas por meio de um processO de fuzzyficação (de "fuzzy" - nebuloso), em proposições nebulosas. As proposições nebulosas são então utilizadas junto com as regras,

para gerar novas proposições nebulosas por meio da inferência nebulosa. Estas novas proposições nebulosas são então transformadas por meio de um processo de defuzzyficação, em sinais de controle que são então enviados aos atuadores do sistema físico. Têm-se portanto a malha de controle mostrada na figura 4.1.

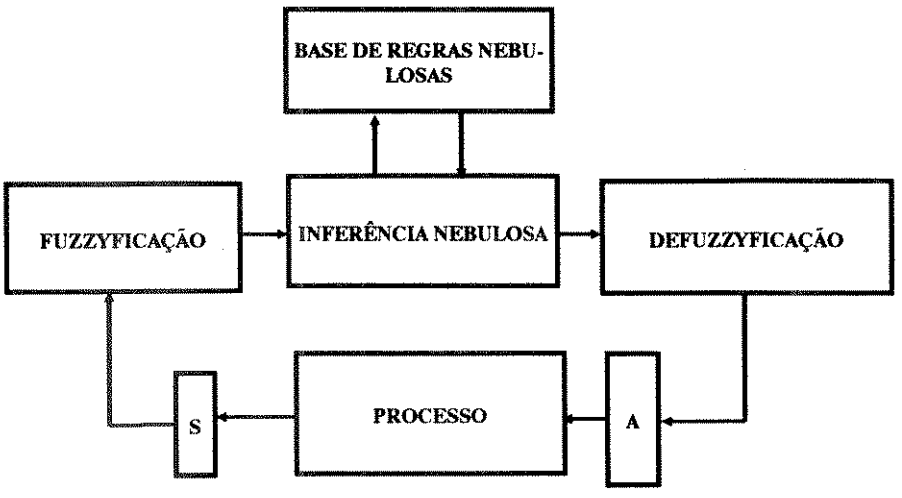


FIGURA 4.1 - Malha de Controle para um Sistema Nebuloso

A vantagem do uso de regras nebulosas ao invés das tradicionais regras de produção é que a semântica das regras nebulosas é mais próxima às construções mentais humanas que as proposições simples. Ou seja, o ser humano está mais habituado a lidar com conceitos, inexatos, imcompletos, imprecisos, do que com proposições booleanas associadas a verdade/não verdade. Assim, fica muito mais fácil a etapa da aquisição do conhecimento do especialista, que para evidenciar seu conhecimento sobre o sistema, terá muito mais facilidade de se expressar podendo utilizar conceitos nebulosos. Com o uso destes, usualmente reduz-se também o número de regras necessárias para descrever o sistema visto que não é necessário o detalhamento minucioso do sistema (implicando na existência de diversas regras a mais para esta finalidade). Um conceito tem uma semântica mais abrangente que uma proposição do tipo verdadeiro/falso, proporcionando uma descrição muito mais sintética.

## 4.4 Fuzzyficação

Para sistemas de controle, o processo de fuzzyficação, é a etapa responsável pela transformação de um sinal vindo de um sensor, que colhe alguma variável do processo, em uma proposição nebulosa do tipo $x_i$ é $A_i$.

Existem vários métodos pelos quais se pode fuzzyficar uma informação vinda do processo. A maneira mais simples de fazê-lo, no entanto, é utilizar uma variável linguística relacionada ao escopo da informação vinda do sensor, e atribuir a ela um conjunto nebuloso formado por um conjunto unitário no ponto referente ao valor enviado pelo sensor.

Suponha-se que o sensor $v_i$ enviou um sinal $mv_i$. Chama-se também de $v_i$ a variável linguística que irá compor a proposição nebulosa referente a esta informação, e $V'_i$ oconjunto nebuloso que irá caracterizar a variável linguística $v_i$, conforme a figura 4.2.


FIGURA 4.2 - Esquema Simplificado de um Fuzzyficador

O conjunto nebuloso V' é caracterizado por sua função de pertinência. Supondo que mv é um elemento do universo de discurso U, mv e U, têm-se:

$$\mu_{V'_i}(u) = \begin{cases} 1, \text{ se } u = mv_i \\ 0, \text{ caso contrário} \end{cases}$$

Transforma-se o assim a informação vinda do sensor $v_i$, $mv_i$, em uma proposição nebulosa "$v_i$ é $V_i$'", a qual pode ser utilizada pelo procedimento de inferência para deduzir os sinais de controle devidos, pois passou-se do domínio numérico (representado por valores colhidos de sensores), para o domínio nebuloso (representado por conceitos associados aos valores colhidos).

## 4.5 Defuzzyficação

Para que uma determinada informação seja utilizada em um sistema de controle, é necessário tomar o resultado da inferência das diversas regras (que serão conjuntos nebulosos) e transformá-lo em valores numéricos correspondentes aos sinais de controle associados às variáveis linguísticas utilizadas nas proposições nebulosas inferidas. Esta etapa é chamada de defuzzyficação, onde a partir de um conjunto nebuloso inferido associado a uma proposição nebulosa, chega-se a um valor associado a um sinal de controle, obtido após o processamento da lógica de decisão.

Um esquema simplificado do processo de defuzzyficação pode ser observado na figura 4.3 a seguir.


FIGURA 4.3 - Esquema Simplificado de um Defuzzyficador

Também no caso da defuzzyficação, várias estratégias podem ser utilizadas. Uma das mais populares é a do centro de massa: calcula-se a integral da função de pertinência inferida, definida sobre o universo de discurso da mesma, e toma-se o ponto que divide o valor desta integral na metade. Supondo-se uma função de pertinência discretizada genérica $\mu_{C'_i}(w^i)$, onde $w^i \in W^i$, sendo $W^i$ o universo de discurso referente à variável linguística $w_i$, o ponto de centro de massa $w_0^i$ é obtido por:

$$w_0^i = \frac{\sum_{j=0}^{q_i} \mu_{C'_i}(w_j^i) \cdot w_j^i}{\sum_{j=0}^{q_i} \mu_{C'_i}(w_j^i)}$$

onde

$q_i$ = número de níveis de quantificação do universo de discurso $W^i$ e

$i = 1,...,m$ ; $m$ = número de variáveis de controle

Após a defuzzyficação, têm-se para cada variável de controle $w^i$ um valor $w_0^i$ que é enviado para o i-ésimo atuador do sistema.

## 4.6 Inferência Nebulosa

A inferência nebulosa é realizada de modo que, a partir de uma série de regras nebulosas e de proposições nebulosas, possa-se deduzir outras proposições nebulosas e com elas, após a defuzzyficação, gerar os sinais de controle.

Para tal, utiliza-se a regra de inferência "Modus Ponens Generalizado", do seguinte modo:

Seja uma regra nebulosa do tipo:

$$\text{SE } (x_1 \text{ é } A_1) \text{ E } (x_2 \text{ é } A_2) \text{ E } \ldots \text{ E } (x_n \text{ é } A_n) \text{ ENTÃO } (y \text{ é } C)$$

Sejam as proposições nebulosas:

- $x_1$ é $A'_1$
- $x_2$ é $A'_2$

...

- $x_n$ é $A'_n$

Deduz-se então a proposição nebulosa y é C' onde C' é dado por:

$$C' = (A'_1 \text{ and } \ldots \text{ and } A'_n) \bullet [(A_1 \text{ and } \ldots \text{ and } A_n) \rightarrow C]$$

$$C' = (A'_1 \times \ldots \times A'_n) \bullet [A_1 \times \ldots \times A_n \times C]$$

$$\mu_{C'}(w) = \bigvee_{(u_1, \ldots, u_n)} \{ [\mu_{A'_1}(u_1) \wedge \ldots \wedge \mu_{A'_n}(u_n)] \wedge [\mu_{A_1}(u_1) \wedge \ldots \wedge \mu_{A_n}(u_n) \wedge \mu_C(w)] \}$$

sendo $(u_1, \ldots, u_n)$ pertencente a $U_1 \times \ldots \times U_n$

Toma-se portanto o supremo, dentre todas as possibilidades de $(u_1, \ldots, u_n)$, da equação seguinte:

$$\mu_{A'_1}(u_1) \wedge \ldots \wedge \mu_{A'_n}(u_n) \wedge \mu_{A_1}(u_1) \wedge \ldots \wedge \mu_{A_n}(u_n) \wedge \mu_C(u_1)$$

Como as funções $\mu_{A'_i}(u)$ são iguais a 0 para todo $u_i \neq u^0_i$, onde $u^0_i$ corresponde ao valor da medida efetuada, para a i-ésima proposição nebulosa (conforme a estratégia de fuzzyficação adotada) e $\mu_{A'_i}(u^0) = 1$, para todos os conjuntos A' (ou $A'_i$), o supremo será sempre o valor correspondA expressão é então simplificada para:

$$\mu_{C'}(w) = \mu_{A_1}(u^0_1) \wedge \ldots \wedge \mu_{A_n}(u^0_n) \wedge \mu_C(u_1)$$

Se $\mu^*$ (chamado de ponto de corte) é definido da seguinte forma:

$$\mu^* = \mu_{A_1}(u^0_1) \wedge \ldots \wedge \mu_{A_n}(u^0_n)$$

têm-se que a função de pertinência inferida da regra é:

$$\mu_{C'}(w) = \mu^* \wedge \mu_C(w)$$

Este esquema de inferência nebulosa é aplicado à todas as regras da base de regras nebulosa, gerando diversos conjuntos nebulosos inferidos. A etapa seguinte é realizar a integração dos conjuntos nebulosos referentes à mesma variável linguística. Ou seja, após a etapa de inferência regra a regra, têm-se uma série de proposições nebulososas do tipo:

$$y_1 \text{ é } C'_4, y_2 \text{ é } C'_2, \ldots, \text{etc.}$$

Algumas proposições nebulosas serão certamente referentes a uma mesma variável linguística, como por exemplo:

$$y_4 \text{ é } C'_2,\ y_4 \text{ é } C'_5,\ y_4 \text{ é } C'_8, \ldots, \text{etc.}$$

Neste caso, é necessário promover a integração destas proposições nebulosas, efetuando a união dos conjuntos nebulosos inferidos, resultando em:

$$y_4 \text{ é } \left[ \bigcup (C'_2, C'_5, C'_8, \ldots, \text{etc}) \right]$$

Após a etapa de integração, pode-se dizer que a inferência nebulosa está realizada. Estas proposições nebulosas poderão então ser transformadas em sinais de controle por meio da defuzzyficação.

## 4.7 Resumo

Neste capítulo, alguns conceitos fundamentais sobre conjuntos nebulosos e lógica nebulosa foram apresentados. Estes conceitos serão utilizados no próximo capítulo para justificar o procedimento de inferência nebuloso via matrizes C-D nebulosas.

# 5. PROCEDIMENTO DE INFERÊNCIA NEBULOSA VIA MATRIZES C-D

## 5.1 Introdução

Neste capítulo mostrar-se-á como o procedimento de inferência via matrizes C-D pode ser extendido de modo a implementar uma máquina de inferência dentro do contexto da lógica nebulosa.

Apesar de sua maior complexidade, a lógica nebulosa de certa forma guarda diversas semelhanças com a lógica proposicional. Analisada sobre uma ótica estrutural, poderia-se até mesmo dizer que a lógica proposicional é um caso particular da lógica nebulosa, onde as funções de pertinência que definem os valores verdade de proposições teriam sempre como contradomínio o conjunto $\{0,1\}$.

Do mesmo modo, o procedimento de inferência via matrizes C-D utilizado na inferência proposicional, pode ser visto como um caso particular de um procedimento de inferência matricial nebuloso.

De fato, a seguir será apresentado um procedimento matricial, chamado de procedimento de inferência nebulosa via matrizes C-D. Como será colocado posteriormente, algumas fases da inferência nebulosa serão delegadas às etapas de pré-processamento e pós-processamento, de modo a compatibilizar o método com o procedimento de inferência via matrizes C-D utilizado na inferência proposicional. Assim o procedimento de inferência via matrizes C-D será considerado como um método único, podendo ser empregado tanto no âmbito da lógica proposicional como no da lógica nebulosa.

## 5.2 Estrutura Geral de Inferência

O procedimento de inferência via matrizes C-D para lógica proposicional convencional, considera uma fase de pré-processamento que colhe as informações do meio, e as converte para um vetor de fatos. Este vetor de fatos alimenta então a máquina de inferência, gerando o vetor de fatos inferidos. O vetor de fatos inferidos passa a seguir por uma etapa de pós-processamento de modo a gerar os sinais de controle adequados ao processo.

Para o caso de sistemas de produção nebulosos, a inferência em si é um pouco mais elaborada. Apesar disso, pode-se adotar a mesma sequência

utilizada para lógica proposicional, ou seja, o pré-processamento/máquina de inferência/pós-processamento. A diferença reside no fato de que as etapas de pré-processamento e pós-processamento fornecerão serviços que conceitualmente estão relacionados ainda à inferência nebulosa. Visto com rigor, o certo seria dizer que o procedimento de inferência nebulosa via matrizes C-D não realiza a inferência completa, mas apenas parte dela. No geral, a sequência pré-processamento/máquina de inferência/pós-processamento realiza a inferência nebulosa completa, permitindo uma implementação eficiente de um mecanismo de inferência.

A seguir, serão consideradas as alterações necessárias nas matrizes C-D nebulosas, nos vetores de fatos nebulosos e nos operadores matriciais. Então descreveremos as atribuições de cada fase, relacionando-as com as etapas da inferência nebulosa.

## 5.3 As Matrizes C-D Nebulosas

Como já havia sido colocado, as matrizes C-D representam as regras que descrevem o sistema de produção. Se compararmos as regras proposicionais com as regras nebulosas, veremos que a estrutura das mesmas é muito semelhante. A única diferença é que ao invés de proposições ordinárias, teremos proposições nebulosas. Em termos de símbolos, pode-se considerar que uma proposição ordinária é análoga a uma proposição nebulosa.

A estrutura das matrizes C e D nebulosas deve portanto, ser semelhante àquela do procedimento de inferência para lógica proposicional, com a ressalva que as proposições representadas serão proposições nebulosas, e não mais proposições simples. Isso significa que, apesar de conter somente 0's e 1's, as matrizes não serão mais booleanas, ou seja, as operações envolvendo as matrizes operacionais e os vetores de fatos não serão booleanas.

Para o caso do procedimento de inferência, via matrizes C-D nebulosas, ser utilizado para regras com coeficiente de certeza, ao invés de regras nebulosas, a matriz D poderá conter elementos diferentes de 0's e 1's, podendo assumir quaisquer valores entre 0 e 1.

## 5.4 Os Vetores de Fatos Nebulosos

Os vetores de fatos nebulosos, a despeito de sua aparente semelhança com os vetores de fatos na lógica proposicional, guardam algumas diferenças conceituais importantes.

No caso da lógica proposicional, cada elemento do vetor de fatos representa o valor verdade da proposição a ele associada. No caso da lógica nebulosa, ao invés do valor verdade tem-se o análogo grau de pertinência. É justamente um grau de pertinência que é representado por cada elemento do vetor de fatos.

Na seção 4.6 mostrou-se que, dentro das condições colocadas pela estratégia de fuzzyficação utilizada, a inferência nebulosa dependerá de um coeficiente $\mu^*$ (chamado de ponto de corte, ou fator de corte). Este fator de corte dependerá basicamente dos valores das funções de pertinência de cada proposição nebulosa nos pontos de amostragem de cada variável linguística. Esta será pois a informação armazenada nos vetores de fatos.

Por exemplo, suponha-se uma variável linguística temperatura, que tenha como atributos os seguintes conjuntos nebulosos: baixa, média e alta. Suponha-se também que um valor específico de temperatura colhido do meio seja de 37° . Deste modo, as proposições nebulosas que podem ser associadas com estas condições seriam:

- temperatura é baixa
- temperatura é média
- temperatura é alta

Para cada conjunto nebuloso relacionado, toma-se então o grau de pertinência associado com o valor específico. Suponha-se que para "baixa", seja 0.3, Para "média" seja 0.8 e para "alta" seja 0.5. Assim, cada proposição nebulosa, de modo análogo ao procedimento convencional, tem um índice no vetor de fatos nebulosos. O elemento referente a este índice, conterá, como informação, o grau de pertinência referente a grandeza física associada. Relacionando-se a cada proposição nebulosa um índice no vetor de fatos nebuloso:

- temperatura é baixa - 0
- temperatura é média - 1
- temperatura é alta - 2

o vetor de fatos nebulosos associado será

$$x = \begin{bmatrix} 0.3 & 0.8 & 0.5 \end{bmatrix}$$

Com esta representação, permite-se a utilização do procedimento de inferência via matrizes C-D, não somente para regras nebulosas, mas também para regras proposicionais com coeficiente de certeza, conforme será discutido posteriormente.

## 5.5 Os Operadores Matriciais Nebulosos

Os operadores matriciais nebulosos, têm uma estrutura idêntica a dos operadores matriciais utilizados anteriormente para a lógica proposicional. Entretanto, o modelo dos operadores escalares conjuntivos e disjuntivos devem ser alterados de modo a incorporar a estrutura multi-valor da lógica nebulosa. Deste modo, ao invés de representar os operadores escalares booleanos *and* e *or*, os símbolos "$\wedge$" e "$\vee$", que caracterizam os operadores escalares conjuntivo e disjuntivo respectivamente, passarão a representar os operadores "min" e "max".

O conectivo conjuntivo "∧" passa a representar uma conjunção dentro do contexto de operações nebulosas, não mais operações booleanas, representando uma operação de "min", ou seja, a escolha do menor dentre os operandos. O conectivo disjuntivo "∨", passa a representar uma disjunção nebulosa, ou seja, uma operação "max", referente à escolha do maior entre os operandos.

Observe-se que apesar desta modificação no conceito dos operadores escalares, esta modificação não altera o funcionamento do mesmo para o contexto booleano. O operador conjuntivo "min", aplicado a operandos booleanos, fornece resultados semelhantes aos do operador booleano *and*. Do mesmo modo, o operador disjuntivo "max", aplicado a valores booleanos, resulta valores semelhantes aos do operador booleano *or*. Assim, temos então que em lógica nebulosa, utilizam-se operadores matriciais generalizados.

## 5.6 A Fase de Pré-Processamento

Como foi dito antes,a fase de pré-processamento abrangerá as etapas iniciais da inferência nebulosa.

Basicamente, promoverá a fuzzyficação (transformação das informações para o domínio nebuloso), e parte da composição.

Na seção 4.6, vimos que após a fuzzyficação, tem início a composição para a determinação do conjunto inferido C'. Esta composição, devido às estratégias escolhidas na própria fuzzyficação e na escolha dos operadores "and", "•" e "⇒", podia ser reduzida analiticamente à seguinte expressão

$$\mu_{C'}(w) = \mu^* \wedge \mu_C(w)$$

Uma etapa inicial nesta composição, era determinar o ponto de corte $\mu^*$.

Como $\mu^*$ é dado pela função

$$\mu^* = \mu^*\left(\mu_{A_1}(u_1^0), \ldots, \mu_{A_n}(u_n^0)\right)$$

observa-se que é fundamental na inferência a obtenção dos valores das funções de pertinência nos pontos referentes às medidas colhidas do meio:

$$\mu_{A_1}(u_1^0), \ldots, \mu_{A_n}(u_n^0)$$

Pela definição de vetores de fatos nebulosos, vemos que estes são os valores que devem assumir os elementos deste. Assim, o pré-processamento, deve se encarregar de preencher os valores dos vetores de fatos nebulosos, com estes valores nos índices correspondentes às proposições nebulosas que têm os conjuntos $A_1, \ldots, A_n$ como atributos.

Assim, o pré-processamento estará preparando o vetor de fatos para o procedimento de inferência. Este se servirá então dos valores contidos no vetor de fatos para a determinação dos valores de corte $\mu^*$ para as diversas regras.

## 5.7 O Procedimento de Inferência

O procedimento de inferência via matrizes C-D nebulosas, será responsável pela determinação dos $\mu^*$ para cada regra, e com isso, determinar o ponto onde ocorrerá o corte para cada consequente em cada regra. Observa-se que quando a proposição nebulosa se encontra no consequente, o valor que seu elemento no vetor de fatos assume, não é mais o valor da função de pertinência para um ponto específico (amostragem), visto que neste caso nem há um ponto de amostragem. Para estes casos, o valor guardado no vetor de fatos é o máximo valor de $\mu^*$ que tem a proposição como consequente em alguma regra.

Em uma primeira etapa, quando se efetua $y = C \dot{\vee} x$, se encontra os valores de $\mu^*$ para cada regra. Na segunda etapa, quando se faz $xf = D \dot{\wedge} y$, os valores de $\mu^*$ são agregados de modo a preencher os elementos dos vetores de fatos relacionados aos consequentes das regras. Em seguida, o pós processamento deverá tomar os valores colocados nos elementos relativos aos consequentes e complementar a inferência nebulosa, efetuando a união de proposições com a mesma variável linguística e efetuando a defuzzyficação. Assim, a base de regras ficará codificada nas matrizes C e D e, após a aplicação do procedimento de inferência via matrizes C-D nebulosas a um vetor de fatos nebulosos, obter-se-á os pontos de corte relativos às proposições nebulosas apontadas nos consequentes das regras.

É necessário frisar que, a despeito de continuarmos a utilizar os termos vetor de fatos nebulosos de entrada e vetor de fatos inferidos nebuloso, a associação dos valores contidos nestes vetores e seu significado prático não é tão imediata como no caso do procedimento convencional. Isto ocorre pois agora o mesmo é utilizado para implementar uma parte bem definida da inferência nebulosa. Além disso, esta parte existe devido a simplificações analíticas que tornaram-se possíveis devido à escolha dos operadores nebulosos, de modo a obter uma simplificação na inferência nebulosa como algoritmo. Talvez uma utilização mais apropriada fosse a de uma base de regras comprimida, em que o vetor de fatos nebulosos de entrada contivesse somente valores referentes às proposições nebulosas citadas nos antecedentes das regras, e o vetor de fatos inferidos nebuloso, por sua vez, só tivesse valores referentes às proposições nebulosas citadas nos consequentes das regras. Com isso, a informação contida nos valores dos elementos dos vetores de fatos de entrada seriam relativas ao valor da função de pertinência referentes ao ponto da grandeza física colhida, em todos os conjuntos nebulosos citados nas proposições nebulosas que são antecedentes das regras. Já a informação contida nos elementos do vetor de fatos inferidos nebuloso, diria respeito ao ponto de

corte da função de pertinência dos conjuntos nebulosos referentes às variáveis linguísticas, nas proposições nebulosas inferidas.

Assim, vemos que o importante no procedimento de inferência via matrizes C-D é a codificação da base de regras em termos de duas matrizes e, por meio de algumas simples operações matriciais, fornecer os pontos de cortes para todas as proposições nebulosas associadas a grandezas de controle do sistema, possibilitando um futuro pós-processamento que transformará esta informação em grandezas físicas diretamente aplicáveis no controle do sistema.

O valor $\mu^*$ para cada proposição nebulosa de saída é dada pela própria definição de ponto de corte:

$$\mu^* = \mu_{A_1}(u_1^0) \wedge \ldots \wedge \mu_{A_n}(u_n^0)$$

que é baseada nos n antecedentes de uma regra, dados na forma das proposições nebulosas ($u_1$ é $A_1$), ... , ($u_n$ é $A_n$), ligadas conjuntivamente. Entretanto, não necessariamente as regras terão antecedentes semelhantes. A idéia aqui é proporcionar um esquema onde cada linha da matriz C, de maneira análoga ao procedimento convencional, escolha dentre a lista de proposições nebulosas, quais são as que figuram no antecedente da regra, para que ao ocorrer a operação matricial, somente estas figurem na operacionalização da regra. O procedimento de inferência via matrizes C-D nebulosas supre justamente este requisito.

DEFINIÇÃO 5.1 Regra Nebulosa

Define-se uma regra nebulosa como uma regra de produção que utiliza proposições nebulosas do tipo ($u_i$ é $A_i$), onde $u_i$ é uma variável linguística e $A_i$ é um conjunto nebuloso. A regra é formada por um antecedente, representando uma condição, estruturado em termos de uma associação conjuntiva de proposições nebulosas e um consequente, uma outra proposição nebulosa. A estrutura básica de uma regra nebulosa é então a seguinte:

SE ($u_1$ é $A_1$) E ($u_2$ é $A_2$) E ... E ($u_n$ é $A_n$) ENTÃO (y é C)

DEFINIÇÃO 5.2 Base de Regras Nebulosa

Uma base de regras nebulosa é a união de várias regras nebulosas, conforme a definição 5.1

DEFINIÇÃO 5.3 Base de Regras Nebulosa Comprimida

Uma base de regras nebulosa comprimida é uma base de regras nebulosa em que não há encadeamento, ou seja, sendo $A_i$ o conjunto de

proposições nebulosas que compõem o antecedente da i-ésima regra da base e $C_i$ o conjunto de proposições nebulosas (unitário) que conpõem o consequente da i-ésima regra da base, e sendo $A$ o conjunto resultante da união de todos os conjuntos $A_i$ da base de regras, e $C$ o conjunto resultante da união de todos os iconjuntos $C_i$ da base de regras, temos que a intersecção dos conjuntos $A$ e $C$ é o conjunto vazio, ou seja

$$A \cap C = \varnothing$$

Isto significa que nenhuma proposição nebulosa que é antecedente de uma regra poderá ser consequente de uma regra, e vice-versa.

DEFINIÇÃO 5.4 Base de Conhecimento Nebulosa Comprimida

Uma base de conhecimento comprimida é formada pela união de uma base de regras nebulosa comprimida, com dois conjuntos, representados por dois vetores, onde o primeiro têm seus elementos correspondentes às proposições nebulosas citadas nos antecedentes das regras e o segundo têm seus elementos correspondentes às proposições nebulosas citadas nos consequentes das regras da base de regras nebulosa comprimida.

DEFINIÇÃO 5.5 Representação de uma Base de Conhecimento Nebulosa Comprimida

Uma base de conhecimentos nebulosa comprimida pode ser representada, para termos de uma inferência nebulosa, por meio das matrizes C-D nebulosas, representando a base de regras nebulosas comprimida, um vetor de fatos nebulosos de entrada, representando o conjunto de proposições nebulosas citadas nos antecedentes das regras da base comprimida e um vetor de fatos nebulosos inferidos, representando as proposições nebulosas citadas nos consequentes das regras da base comprimida.

DEFINIÇÃO 5.6 Espectro Linguístico

Seja $l_i$ uma variável linguística.

Seja $P^i$ o conjunto de conceitos associados à variável linguística $l_i$ cada conceito representado por meio de um conjunto nebuloso $p^i_j$.

Define-se $E_i$ ,o espectro linguístico de $l_i$ como sendo o produto cartesiano $l_i \times P^i$:

$$E_i = l_i \times P^i = \left\{ (l_i, p^i_1), (l_i, p^i_2), \ldots, (l_i, p^i_{n(i)}) \right\}$$

onde cada elemento do espectro linguístico ( $l_i$ , $p_j^i$ ) está associado a uma proposição nebulosa do tipo $l_i$ é $p_j^i$ e n(i) é a cardinalidade do espectro linguístico $E_i$ .

DEFINIÇÃO 5.7 Procedimento de Inferência Nebuloso

Seja BRN uma base de regras nebulosas comprimida.

Seja $L = \{ l_i \}$ o conjunto de variáveis linguísticas da base BRN, associadas às grandezas $f_i$ .

Seja $E_i$ o espectro linguístico da variável linguística $l_i$ . Seja $L^A$, $L^A \subset L$, o conjunto das variáveis linguísticas $l_i$ que constam somente dos antecedentes das regras.

Seja $L^C$, $L^C \subset L$, o conjunto das variáveis linguísticas $l_i$ que constam somente nos consequentes das regras.

Seja $P^A$ o conjunto resultante da união dos espectros linguísticos das variáveis linguísticas contidas em $L^A$, contendo todas as proposições nebulosas que são citadas nos antecedentes das regras.

Seja $P^C$ o conjunto resultante da união dos espectros linguísticos das variáveis linguísticas contidas em $L^C$, contendo todas as proposições nebulosas que são citadas nos consequentes das regras.

Seja x o vetor de fatos de entrada nebuloso, onde cada elemento de x representa uma proposição nebulosa de $P^A$ sendo que o número de elementos de x é igual à cardinalidade de $P^A$ .

Seja xf o vetor de fatos inferidos nebuloso, onde cada elemento de xf Crepresenta uma proposição nebulosa de $P^C$ sendo queo número de elementods de xf é igual à cardinalidade de $P^C$ .

Sejam C e D as matrizes operacionais nebulosas comprimidas que representam uma base de regras nebulosas comprimida.

Seja $\overline{C}$ a matriz C negada elemento a elemento: $\overline{c}_{ij} = 1 - c_{ij}$

Sejam $\dot{\wedge}$ e $\dot{\vee}$ os operadores matriciais nebulosos.

Define-se o procedimento de inferência nebuloso como a seguinte sequência de operações:

$$xf = D \dot{\wedge} (\overline{C} \dot{\vee} x)$$

## 5.8 A Fase de Pós-Processamento

A fase de pós-processamento se encarrega de tomar o vetor de fatos inferidos xf, contendo os valores onde cada proposição nebulosa citada nos consequentes das regras deve ser cortada, e montar efetivamente as funções de pertinência inferidas. Em seguida, deve tomar cada função inferida para cada variável linguística e defuzzyficá-la, gerando o sinal que deve ser efetivamente enviado para o controle do sistema. Este processamento consiste em tomar o vetor de fatos inferidos e para cada variável linguística citada nos consequentes das regras, gerar um valor que corresponderá ao resultado final da inferência, utilizado como sinal de controle.

DEFINIÇÃO 5.8 - Agregação Nebulosa de um Espectro Linguístico

Seja BRN uma base de regras nebulosa comprimida.

Seja xf o vetor de fatos inferidos, resultado da aplicação do procedimento de inferência colocado pela definição 5.7 sobre um vetor de fatos x.

Seja $E_i$ o espectro linguístico de uma variável linguística $l_i$.

Seja $xf^i$ um vetor de fatos formado somente pelos elementos de xf que correspondam às proposições nebulosas relativas ao espectro linguístico $E_i$ de $l_i$.

Define-se a agregação $G_i$ do espectro linguístico $E_i$ de $l_i$ como sendo o conjunto nebuloso resultado da união de todos os conjuntos nebulosos do espectro linguístico $E_i$ cortados pelos fatores especificados em $xf^i$:

$$\mu_{G_i}(w) = \sup_k \left( xf^i[k] \wedge p_k^i(w) \right)$$

DEFINIÇÃO 5.9 - Pós-Processamento Nebuloso

Seja BRN uma base de regras nebulosa comprimida.

Considere-se o procedimento de inferência nebuloso colocado pela definição 5.7, aplicado sobre a base BRN.

O pós-processamento nebuloso que completa a inferência nebulosa é dado pela seguinte sequência:

- Para cada variável linguística de $L^C$, efetue a agregação nebulosa de seu espectro linguístico.
- Para cada agregação gerada no passo anterior, efetue a defuzzyficação, conforme a seção 4.5.

Após estas duas etapas, o pós-processamento nebuloso terá fornecido a inferência nebulosa de forma completa. Os demais serviços que se façam necessários serão particulares da aplicação envolvida.

## 5.9 Teorema da Inferência Nebulosa

TEOREMA 5.1 - Teorema da Inferência Nebulosa

Seja um vetor de fatos nebuloso de entrada, preenchido pelo pré-processamento nebuloso.

Seja o procedimento de inferência via matrizes C-D nebulosas conforme a definição 5.7.

Seja o pós-processamento nebuloso, conforme a definição 5.8.

Então, a inferência nebulosa completa, por "Modus Ponens Generalizado", de todas as variáveis lingüísticas citadas nos consequentes das regras é dada pela aplicação do procedimento de inferência via matrizes C-D nebulosas sobre vetor de fatos nebuloso de entrada, seguido do pós-processamento do resultado obtido.

**PROVA:**

De acordo com a seção 4.6, o procedimento de "Modus Ponens Generalizado" pode ser reduzido à seguinte expressão, para uma regra nebulosa $R_i$:

$$\mu_{Q'_i}(w) = \mu_i^* \wedge \mu_{Q_i}(w)$$

onde $\mu_i^* = \mu_{A_{i1}}(u_1^0) \wedge \ldots \wedge \mu_{A_{in}}(u_n^0)$,

$A_{il}$ é o l-ésimo antecedente da regra $R_i$ e

$Q_i$ é o consequente da regra $R_i$.

$Q'_i$ é o conjunto nebuloso inferido por "Modus Ponens Generalizado" pela regra $R_i$

Pela definição do operador matricial disjuntivo, temos que

$$y = \overline{C} \dot{\vee} x = \left\{ y_i \mid y_i = \bigwedge_j (\overline{c}_{ij} \vee x_j) \right\}$$

O valor de $y_i$ passa então a ser a produtória booleana

$$y_i = (\overline{c}_{i1} \vee x_1) \wedge (\overline{c}_{i2} \vee x_2) \wedge \ldots \wedge (\overline{c}_{in} \vee x_n)$$

Pela definição da estrutura da matriz C, temos que $c_{ij} = 1$ se a entidade $e_j$ (correspondente a uma proposição nebulosa) pertence ao antecedente da regra $R_i$, sendo 0 caso contrário. Se $c_{ij} = 0$, então $\bar{c}_{ij} = 1$. Para estes casos, a parcela $(\bar{c}_{ij} \vee x_j)$ será 1, qualquer que seja o valor de $x_j$. Deste modo, o valor desta parcela não irá influir no valor do produtório. Supondo então um conjunto $K_i^C$, formado pelos índices k tais que $\bar{c}_{ik} = 1$, e um conjunto $L_i^C$, formado pelos índices l tais que $\bar{c}_{il} = 0$, podemos então dividir o produtório em duas partes:

$$y_i = \left( \bigwedge_k (\bar{c}_{ik} \vee x_k) \right) \wedge \left( \bigwedge_l (\bar{c}_{il} \vee x_l) \right)$$

onde $\bar{c}_{ik}$ é sempre 1 $\forall k \in K_i^C$ e $\bar{c}_{il}$ é sempre 0 $\forall l \in L_i^C$.
Como $(\bar{c}_{ik} \vee x_k)$ é sempre igual a 1, $\forall k \in K_i^C$, temos que

$$\left( \bigwedge_k (\bar{c}_{ik} \vee x_k) \right) = 1$$

e então

$$y_i = \left( \bigwedge_l (\bar{c}_{il} \vee x_l) \right)$$

Do mesmo modo, como $\bar{c}_{il}$ é sempre igual a 0, $\forall l \in L_i^C$, temos que:

$$y_i = \bigwedge_l x_l$$

Se $\bar{c}_{il} = 0$, $c_{il} = 1$ então $e_l$ pertence ao antecedente da regra $R_i$, $\forall l \in L_i^C$. Pela definição de vetor de fatos nebuloso, um elemento $x_l$ deve conter como valor o grau da função de pertinência correspondente ao ponto de amostragem. Ou seja $x_l = \mu_{A_{il}}(u_l^0)$. Se $e_l$ pertence ao antecedente da regra $R_i$, , $\forall l \in L_i^C$, então o valor de $y_i$ é exatamente igual ao valor $\mu_i^*$:

$$\mu_i^* = y_i = \bigwedge_l x_l = \bigwedge_l \mu_{A_{il}}(u_l^0) = \mu_{A_{i1}}(u_1^0) \wedge \ldots \wedge \mu_{A_{in}}(u_n^0)$$

Em seguida, de acordo com a seção 4.6, para concluir a inferência nebulosa, promove-se a união dos conjuntos nebulosos referentes à mesma variável linguística, obtidos nas diferentes regras:

$$vl_n \text{ é S, onde } S = \bigcup_v (Q'_v) \text{ e}$$

$$\mu_S(w) = \mu_{Q'_1}(w) \vee \ldots \vee \mu_{Q'_m}(w), \forall w \in U$$

$vl_n$ é a n-ésima variável linguística de saída,

S é o conjunto nebuloso inferido

$Q'_v$ é o conjunto nebuloso inferido por uma regra $R_v$, que tem em sua proposição nebulosa a variável linguística $vl_n$.

Como $\mu_{Q'_v} = \mu^*_v \wedge \mu_{Q_i}$ tem-se

$$\mu_S(w) = (\mu^*_1 \wedge \mu_{Q_{i1}}) \vee \ldots \vee (\mu^*_m \wedge \mu_{Q_{im}})$$

onde os índices $i_1, \ldots, i_m$ dependerão dos consequentes das regras. Supondo-se alguns índices coincidentes ($i_u = i_v$) tem-se:

$$\mu_S(w) = (\alpha_1 \wedge \mu_{Q_i}) \vee \ldots \vee (\alpha_d \wedge \mu_{Q_d})$$

onde

$$\alpha_i = \left( \bigvee_k \mu^*_k \right) \quad (5.1)$$

Nesta fórmula, a somatória nebulosa é varrida $\forall k \in K_i^D$, onde $K_i^D$ é o conjunto de regras que tem $Q_i$ como consequente (e por conseguinte, tem seus elementos na matriz D ($d_{ik}$) iguais a 1).

Pela definição do operador matricial conjuntivo, temos:

$$xf = D \dot{\wedge} y = \left\{ xf_i \mid xf_i = \bigvee_j (d_{ij} \wedge y_j) \right\}$$

Pela definição da estrutura da matriz D, temos que $d_{ij} = 1$ se $e_i$ pertence ao consequente da regra $R_j$ e 0 em caso contrário. Supondo então um conjunto $K_i^D$, formado pelos índices k tais que $d_{ik} = 1$, e um conjunto $L_i^D$, formado pelos

índices l tais que $d_{il} = 0$, podemos então dividir a somatória booleana em duas partes:

$$xf_i = \left( \bigvee_k (d_{ik} \wedge y_k) \right) \vee \left( \bigvee_l (d_{il} \wedge y_l) \right)$$

onde $d_{ik}$ é sempre 1, $\forall k \in K_i^D$ e $d_{il}$ é sempre 0, $\forall l \in L_i^D$ .

Se $d_{il}$ é sempre 0, a parcela $(d_{il} \wedge y_l)$ será sempre 0, qualquer que seja o valor de $y_l$ , $\forall l \in L_i^D$ .

Com isso, temos que

$$\bigvee_l (d_{il} \wedge y_l) = 0$$

e com isso

$$xf_i = \left( \bigvee_k (d_{ik} \wedge y_k) \right)$$

Como $d_{ik}$ também é sempre igual a 1, $\forall k \in K_i^D$, temos que:

$$xf_i = \bigvee_k y_k \quad \text{e como} \quad y_k = \mu_k^*$$

$$xf_i = \bigvee_k \mu_k^*$$

que equivale a (5.1), ou seja, $xf_i = \alpha_i$. Deste modo, a aplicação do procedimento de inferência via matrizes C-D nebulosas, vai nos gerar os valores $\alpha_i$ que serão utilizados para completar a inferência nebulosa.

Para concluir a inferência nebulosa, falta finalizar a operação:

$$\mu_S(w) = (\alpha_1 \wedge \mu_{Q_1}) \vee \ldots \vee (\alpha_d \wedge \mu_{Q_d}) \quad (5.2)$$

Pela definição do pós-processamento nebuloso, a primeira etapa do mesmo corresponde à determinação da agregação nebulosa para cada variável linguística de saída:

$$\mu_G(w) = \sup_k \left( xf^i[k] \wedge p_k^i(w) \right) \quad (5.3)$$

De acordo como as suposições da definição da agregação linguística, i corresponde ao índice da variável linguística. Neste contexto $\mu_S$, deve ser entendido então como $\mu_{Si}$, pois também corresponde a apenas uma variável linguística. Esta fórmula é exatamente igual a (5.2), observando-se que G = S, $xf^i[k] = \alpha_k$ e $p_k^i(w) = Q_k(w)$

Deste modo, conclui-se que a aplicação do procedimento de inferência via matrizes C-D nebulosas seguida da agregação nebulosa dos espectros linguísticos realiza a inferência nebulosa completa. Como o pós-processamento nebuloso colocado pela definição 5.9 inclui a agregação linguística, isto conclui a prova do teorema 5.1.

O processo de defuzzyficação, colocado a mais no pós-processamento, somente é necessário para a utilização prática dos conjuntos nebulosos inferidos por "Modus Ponens Generalizado", fazendo um mapeamento entre o "mundo nebuloso" e o mundo real.

## 5.10 Resumo

Neste capítulo, descreveu-se como se utilizar o procedimento de inferência via matrizes C-D em aplicações envolvendo lógica nebulosa. Os significados dos vetores de fatos, matrizes operacionais e operadores matriciais foram definidos dentro de um contexto mais abrangente, de modo a implementar um mecanismo de inferência de uso geral, podendo ser usado do mesmo modo tanto em sistemas baseados em lógica proposicional como em sistemas que utilizam a lógica nebulosa.

No capítulo seguinte será feita uma análise de desempenho do procedimento de inferência via matrizes C-D, em suas diversas formas de implementação, bem como uma comparação com outros tipos de máquina de inferência.

# 6. IMPLEMENTAÇÃO COMPUTACIONAL E ANÁLISE DE DESEMPENHO

## 6.1 Introdução

Neste capítulo serão desenvolvidas diversas estratégias de implementação do procedimento de inferência via matrizes C-D, detalhado nos capítulos 2 e 3, bem como sua generalização abrangendo sistemas nebulosos, colocada no capítulo 5.

Como será verificado, para uma implementação computacional eficiente do algoritmo [1], é necessário considerar alguns aspectos estruturais da base de conhecimento, bem como o tipo de máquina (hardware) em que a mesma será implementada, de modo que seja adotada uma estrutura que minimize os cálculos computacionais para uma determinada aplicação, ou grupo de aplicações. Nestas, adotam-se restrições na estrutura do conhecimento de modo a caracterizar a implementação computacional mais adequada.

## 6.2 Implementação de Vetores e Matrizes

O procedimento de inferência via matrizes C-D generalizado (abrangendo tanto o caso da lógica proposicional clássica como o da lógica nebulosa), adota uma representação que especifica matrizes e vetores, que serão manipulados a fim de gerar a inferência sobre a base de conhecimento que tais matrizes representam. Na implementação prática do procedimento, tais vetores e matrizes devem ser implementados em termos de uma estrutura computacional.

De acordo com o exposto nos capítulos 2 e 5, as matrizes C e D são sempre matrizes binárias, ou seja, matrizes cujos elementos são sempre 0 ou 1. No caso dos vetores utilizados, sendo o sistema implementado dentro do âmbito da lógica proposicional clássica, os mesmos serão também binários. No caso da lógica nebulosa, os vetores não são binários, podendo assumir valores no intervalo [0,1].

Tanto as matrizes como os vetores binários permitem vários tipos de implementação.

Analisemos o caso de um vetor binário. Pode-se implementar o mesmo gerando um *array* de memória com tantos elementos quantos forem os elementos do vetor e, dando a cada um destes, o mesmo valor (0 ou 1) que seu correspondente no vetor. Esta é a implementação mais imediata. Outra maneira de se implementar este vetor, é gerar um *array* de memória e para cada elemento deste array colocar, como valor, o índice para o qual o vetor têm valor 1. Para este caso, o *array* utilizado para a implementação não terá uma dimensão fixa, pois a mesma dependerá do número de 1's do vetor. Pode-se entretanto considerar uma dimensão máxima, que é quando todos os elementos do vetor forem 1's, quando o *array* terá a mesma dimensão do vetor. Para descrever a dimensão útil do *array*, será necessário uma outra referência. Ou se guarda esta dimensão no valor de uma variável, diretamente relacionada ao vetor, ou utiliza-se um caracter especial que represente fim de vetor (um número negativo, por exemplo), de modo que o mecanismo que efetua as operações matriciais saiba se está ainda lendo o *array* ou não. Uma implementação que é equivalente a esta é colocar como valor dos elementos do *array* os índice para os quais o vetor têm valor 0. Estas implementações são exemplificadas na tabela 6.1.

VETOR - [ 0, 0 ,1, 0, 1 ]

| | 1º byte | 2º byte | 3º byte | 4º byte | 5º byte | |
|---|---|---|---|---|---|---|
| TIPO $\alpha$ | 0 | 0 | 1 | 0 | 1 | (6.1) |
| TIPO $\beta$ | 2 | 2 | 2 | - | - | (6.2) |
| TIPO $\gamma$ | 2 | 4 | -1 | - | - | (6.3) |
| TIPO $\delta$ | 3 | 0 | 1 | 3 | - | (6.4) |
| TIPO $\varepsilon$ | 0 | 1 | 3 | -1 | - | (6.5) |

**Tabela 6.1 - Exemplos de Implementação de Vetores Binários**

A implementação colocada em (6.1), doravante chamada de tipo $\alpha$, corresponde à implementação direta do vetor. A implementação (6.2), que será chamada de tipo $\beta$, coloca no 1º byte o número de elementos do vetor com valor 1 e em seguida os índices correspondentes. A implementação (6.3) (TIPO $\gamma$) coloca os elementos do vetor com valor 1, finalizando a série com uma marca em -1, indicando o fim do vetor. Em (6.4) (TIPO $\delta$) ocorre o mesmo que em (6.2), mas com os elementos do vetor de valor 0, e em (6.5) (TIPO $\varepsilon$) ídem em relação a (6.3).

Uma matriz pode ser interpretada como a junção de diversos vetores, em linha ou em coluna. No caso de junção em linha, cada linha da matriz será implementada por um vetor, que por sua vez poderá ser implementado por um dos cinco tipos já apresentados. Sendo interpretada como a junção de colunas a implementação é semelhante. Normalmente utiliza-se a mesma implementação para todos os vetores, tanto para o caso do tipo linha como do tipo coluna. A escolha da implementação é caracterizada pelo tipo de representação de vetor utilizada e pelo tipo de junção, se de linhas ou colunas. Tem-se então 10 tipos diferentes de implementação de matriz:

| $\alpha$-linha | $\beta$-linha | $\gamma$-linha | $\delta$-linha | $\varepsilon$-linha |
|---|---|---|---|---|
| $\alpha$-coluna | $\beta$-coluna | $\gamma$-coluna | $\delta$-coluna | $\varepsilon$-coluna |

Além destes tipos de implementação de vetores e matrizes, existem implementações híbridas, que podem particionar o vetor/matriz, e implementar cada partição por um dos tipos apresentados. Este tipo de técnica pode ser interessante, principalmente quando se tratar de casos de bases de conhecimento híbridas, contendo tanto proposições clássicas como nebulosas, que podem ter o desempenho de sua implementação incrementado com o uso de tais técnicas, desde que se faça uma partição adequada da base de conhecimento.

A implementação de vetores e matrizes não binárias somente poderá ser do tipo $\alpha$, no caso de vetores,ou $\alpha$-linha e $\alpha$-coluna para o caso de matrizes.

A escolha da implementação mais adequada de implementação está intimamente ligada ao tipo de processamento que será utilizado, se em máquinas sequenciais ou máquinas paralelas. Esta escolha pode também ser melhor avaliada, caso se conheça peculiaridades da base de conhecimento a ser empregada. Estas considerações serão colocadas nas seções seguintes, onde as implementações que mostram-se mais adequadas em cada caso serão discutidas.

## 6.3 Desempenho em Máquinas Sequenciais

Em máquinas sequenciais (*hardware* sequencial), cada operação do procedimento de inferência, será efetuada em um instante de tempo distinto, ou seja, não se admite que se façam operações em paralelo. Para encontrar o melhor tipo de implementação para estes tipos de máquina, é necessário analisar-se os operadores matriciais e como os mesmos agirão diante das matrizes C e D utilizadas, bem como o vetor de fatos.

Utilizando-se uma base de regras expandida horizontalmente, o ciclo de inferência se reduzirá à aplicação única de um passo de inferência. Isto facilita o cálculo do tempo de inferência.

Analisemos as operações necessárias para implementar a primeira parte do procedimento, ou seja, a operação $y = C \dot{\wedge} x$:

Existem diversas maneiras de implementar um operador de matrizes. Entretanto, é necessário seguir uma sequência, que não pode ser modificada. Para o caso do operador $\dot{\vee}$ é necessário antes de se executar a operação $\wedge$ da linha, efetuar a operação $\vee$ com o vetor operado. Na prática isto significa que para um elemento qualquer da matriz, de índices i e j, é necessário antes se fazer a operação escalar $p_{ij} = \bar{c}_{ij} \vee x_j$ para depois se utilizar o termo $p_{ij}$ na produtória booleana, fazendo $y_i = y_i \wedge p_{ij}$. Seguindo-se esta regra, a escolha dos índices i e j utilizados pode ser qualquer. Isto nos dá uma grande flexibilidade na implementação do operador matricial. Observe-se que, se fossemos utilizada uma ordem totalmente aleatória na escolha dos índices, seria necessário armazenar os valores $p_{ij}$, o que implicaria o dobro de memória. Isso

pode ser evitado, fazendo-se que logo após o cálculo de $p_{ij}$, o mesmo já seja incorporado à produtória booleana. Deste modo, pode-se utilizar uma variável auxiliar única para guardar o valor de $p_{ij}$, já que o mesmo será utilizado logo em seguida (o que é feito uma única vez para cada termo), livrando a variável para um novo cálculo. Deste modo, uma implementação eficiente do operador deve ter como cálculos consecutivos as seguintes operações escalares:

$$aux = \overline{c}_{ij} \vee x_j$$
$$y_i = aux \wedge y_i$$

ou, de um modo mais conciso: $y_i = y_i \wedge (\overline{c}_{ij} \vee x_j)$;

Seguindo-se esta regra, esta operação pode ser efetuada para os índices i e j de qualquer forma, o que libera ao programador a escolha da ordem mais apropriada. Entretanto, a princípio (sem se levar em conta as estruturas da matriz C, do vetor x e vetor y), todos os índices devem ser varridos, de modo que o valor de $y_j$ seja o valor verdadeiro (só a ordem que pode ser qualquer).

A maneira mais simples de se implementar tal sequência de operações é através de um loop controlado que varra todos os valores de i e j, o que pode ser feito, por exemplo, das seguintes maneiras (descritas a seguir em pseudo-código):

```
for (i=0;i <m;i++)
      y[i] = 1;
for (j=0;j <n;j++)
      for (i=0;i<m;i++)
                  y[i] = y[i] ∧ (C̄[i][j] ∨ x[j]);
```

ou

```
for (i=0;i <m;i++)
      {y[i] = 1;
       for (j=0;j <n;j++)
                  y[i] = y[i] ∧ (C̄[i][j] ∨ x[j]);
      }
```

Outra maneira de se implementar o mesmo procedimento é, ao invés de se utilizar os índices de i e j sequencialmente, colocá-los em duas listas: as listas dos índices i's e a lista dos índices j's, onde a ordem em que o mesmo será processado é a ordem de sua aparição na lista. Suponha-se uma matriz C com número de linhas 5 e número de colunas 3. Montam-se então duas listas, com os índices que serão processados, por exemplo:

```
l = [ 5,3,2,4,1 ]
c = [ 1,3,2 ]
```

Pode-se então iniciar o processamento pela linha ou pela coluna. Caso se escolha a linha, processar-se-ão os seguintes pares de (i,j):

(5,1) , (5,3) , (5,2) , (3,1), (3,3), ...

cujo procedimento em pseudo-código seria, para m = 5 e n = 3:

```
for (i=0;i <m;i++)
    {y[ l[i] ] = 1;
     for (j=0;j<n;j++)
                y[ l[i] ] = y[ l[i] ] ∧ (C̄ [ l[i] ][ c[j] ] ∨ x[ c[j] ]);
    }
```

Caso se iniciasse pela coluna, teriam-se os seguintes pares:

(1,5) , (1,3) , (1,2), (1,4) , (1,1) , (3,5) , (3,3) , ...

cujo procedimento em pseudo-código seria:

```
for (i=0;i <m;i++)
      y[ l[i] ] = 1;
for (j= 0;j <n;j++)
      for (i=0;i<m;i++)
                y[ l[i] ] = y[ l[i] ] ∧ (C̄[ l[i] ][ c[j] ] ∨ x[ c[j] ]);
```

Outra variação deste algoritmo seria fornecer uma lista l que contivesse os índices das linhas a serem executadas, e uma matriz c, que para cada linha contivesse os índices das colunas, permitindo um sequenciamente completamente independente, do processamento dos termos da matriz. Pode-se também fazer o contrário, ou seja, uma lista c dando a ordem dos índices das colunas, e uma matriz l, que para cada coluna daria a ordem do processamento dos índices das linhas.

Este tipo de implementação nos dá margem a algumas manipulações. Inicialmente verifica-se que ele permite alterar o sequenciamento das operações, independente da constituição interna da matriz. Caso analisemos esta constituição interna, podemos mesmo suprimir algums índices, cujas operações sabidamente não terão efeito no resultado final.

Observando-se a fórmula que é executada, verifica-se que o valor de $y_i$ só é modificado, quando o valor de $\bar{c}_{ij}$ é igual a zero. Desta forma, quando este valor é 1, o resultado será o mesmo, qualquer que seja o valor de $x_j$. Para aumentar o desempenho da máquina de inferência, para o caso sequencial, seria possível que não se processasse a equação para os valores de i e j tais que $\bar{c}_{ij}$ seja 1, pois estas parcelas não contribuem em nada para o resultado.

-Do mesmo modo, quando $\bar{c}_{ij}$ é igual a 0, a operação $\bar{c}_{ij} \vee x_j$ é igual a $x_j$, podendo portanto ser omitida. Pode-se então, desde que se utilizem apenas os valores corretos de i e j, efetuar simplesmente a operação:

$$y_i = y_i \wedge x_j$$

Uma maneira de se ter somente os índices i e j em que $\bar{c}_{ij} = 0$, é utilizar uma nova matriz que seja implementação da matriz C como do tipo $\delta$-linha, $\delta$-coluna, $\varepsilon$-linha ou $\varepsilon$-coluna. Nestas implementações, os índices que correspondem a valores de $\bar{c}_{ij}$ iguais a 1 não são nem citados. Com isso, evita-se gastar tempo processando elementos que não influenciam no resultado final. Implementações do tipo linha, deverão ser varridas para todas as m linhas, mas terão então como variáveis de controle dos *loops* de coluna, não n, mas n'[i], onde n'[i] $\leq$ n é o número de elementos da linha i com valores iguais a 0. No caso de uma implementação do tipo $\delta$, n'[i] é dado pelo primeiro elemento da linha. No caso do tipo $\varepsilon$, ele é dado contando-se os termos até encontrar a marca de final de linha. Implementações do tipo coluna deverão ser varridas por todas suas n colunas, mas serão varridas nas linhas apenas até m'[j], onde m'[j] $\leq$ m é o número de elementos da coluna j com valor igual a 1. Esta implementação poupa então tempo de execução e espaço de memória.

Supondo a implementação da matriz $\bar{C}$ por meio de um *array* *CL*, de tipo $\delta$-linha, teremos o seguinte algoritmo:

```
for (i=0;i <m;i++)
    {y[i] = 1;
     for (j=0;j <CL[i][0];j++)
                y[i] = y[i] ∧ x[ CL[i][j+1] ];
    }
```

Este caso é semelhante a ter uma lista l[i] = i, que coloca a ordem das linhas a serem processadas, e a uma matriz c, que é a própria matriz *CL*, que coloca para cada linha, quais as colunas a serem processadas.

Para o caso de se utilizar uma implementação *CC* do tipo $\delta$-coluna, tem-se:

```
for (i=0;i <m;i++)
    y[i] = 1;
for (j= 0;j <n;j++)
    for (i=1;i<CC[j][0] ;i++)
                y[ CC[j][i+1] ] = y[ CC[j][i+1] ] ∧ x[j];
```

Outra possibilidade de se evitar processamento desnecessário, ocorre quando o valor de $y_i$ torna-se 0. Neste caso, o valor de $y_i$ não poderá mais ser modificado (devido à produtória booleana). Assim, pode-se fazer um teste no *loop* de controle que envolve a fórmula, de modo que se evite o cálculo da mesma, quando o valor de $y_i$ é 0:

```
for (i=0;i <m;i++)
      {y[i] = 1;
       for (j=0;j <CL[i][0] && y[i] == 1;j++)
                  y[i] = y[i] ∧ x[ CL[i][j+1] ];
      }

for (i=0;i <m;i++)
      y[i] = 1;
for (j= 0;j <n;j++)
      for (i=1;i <CC[j][0] && y[i] == 1;i++)
                  y[ CC[j][i+1] ] = y[ CC[j][i+1] ] ∧ x[j];
```

Estas implementações nem sempre são interessantes, pois aumentam o tempo de processamento de cada ciclo, avaliando o valor de $y_i$. No caso da primeira implementação, entretanto, para os casos em que o valor de $y_i$ torna-se 0 logo nas primeiras iterações, evita-se muito tempo de processamento, sendo que nestes casos, a mesma torna-se uma possibilidade interessante. Ao contrário, a segunda implementação não é muito interessante, pois apesar de não mais necessitar de continuar o *loop* para a variável j (quando se atinge $y_i = 0$), o loop não pode ser finalizado pois o mesmo é externo.

As mesmas considerações efetuadas para a operação $y = \overline{C} \dot{\vee} x$, podem ser extendidas para a operação seguinte que é $xf = D \dot{\wedge} y$. Para o caso da matriz D, entretanto, a fórmula calculada no *loop* interno só não se modifica para os valores de $d_{ij}$ iguais a 0. Do mesmo modo, atingindo um valor igual a 1, uma variável $xf_i$ também não mais se modifica. Com isso, as implementações interessantes para a matriz D passam a ser as do tipo $\beta$ e $\gamma$. Tem-se então os seguintes algoritmos, onde *DL* é a implementação de D do tipo $\beta$-linha e *DC* do tipo $\beta$-coluna:

```
for (i=0;i <n;i++)
      {xf[i] = 0;
       for (j=0;j <DL[i][0] && xf[i] == 0;j++)
                  xf[i] = xf[i] ∨ y[ DL[i][j+1] ];
      }

for (i=0;i <n;i++)
      xf[i] = 0;
for (j= 0;j <m;j++)
      for (i=0;i <DC[j][0] && xf[i] == 0;i++)
                  xf[ DC[j][i+1] ] = xf[ DC[j][i+1] ] ∨ y[j];
```

Para finalizar o procedimento, ou seja fazer com que

$$xf = x + D \dot{\wedge} (\overline{C} \dot{\vee} x)$$

basta encadear os algoritmos para a primeira e para a segunda operação matricial. A soma booleana termo a termo do resultado da operação com o vetor x original, pode ser feita não se zerando o termo xf[i] no segundo algoritmo. A integração dos dois algoritmos pode ser feita de diversas formas, sendo particularmente interessante a seguinte, que permite que não se armazene o vetor intermediário y, transformando-o em uma variável auxiliar simples:

```
for (i=0;i <m;i++)
    {y = 1;
     for (j=1;j <CL[i][0] && y == 1;j++)
                y = y ∧ x[ CL[i][j] ];
     for (j=1;j <DC[i][0] && xf[ DC[i][j] ] == 0;j++)
                xf[ DC[i][j] ] = xf[ DC[i][j] ] ∨ y;
    }
```

Para o caso em que as regras têm apenas um consequente, (ou seja, $DC[i][0] = 1$ para todo i), pode-se simplificar ainda mais o algoritmo, transformando a matriz *DC* em um vetor *DV*, onde cada elemento do mesmo representa para cada coluna de D, a única linha onde $d_{ij} = 1$, em uma implementação do tipo $\gamma$-coluna, onde não há marca de fim de linha, pois a mesma só tem um elemento. Daí resulta:

```
for (i=0;i <m;i++)
    {y = 1;
     for (j=1;j <CL[i][0] && y == 1;j++)
                y = y ∧ x[ CL[i][j] ];
     xf[ DV[i] ] = xf[ DV[i] ] ∨ y;
    }
```

Esta implementação do procedimento de inferência via matrizes C-D tem um tempo de execução bem definido, que é calculado a seguir.

O algoritmo pode ser reescrito da seguinte forma:

```
for (i=0;i <m;i++)
    {y = 1;
     a = CL[i][0];
     for (j=1;j <=a && y == 1;j++)
                {b = x[ CL[i][j] ]
                 y = y ∧ b;
                }
     &b = &xf[ DV[i] ];
     b = b ∨ y;
    }
```

Vamos calcular o tempo de execução da implementação em termos de operações elementares:

- Atribuição
- Soma
- Comparação
- Incremento

Cujos tempos de execução serão computados como A, S, C e I.

Tanto a operação ∧ como ∨ podem ser interpretadas em termos de 1 comparação e 1 atribuição (C+A):

```
min (a,b)
{if (a <b) return(a);
 else return(b);
}

max (a,b)
{if (a >b) return(a);
 else return(b);
}
```

Suponha-se que a média dos valores de *CL*[i][0], para $0 \leq i \leq m$, seja n'. Então os tempos de execução do algoritmo podem ser computados:

```
for (i=0;i <m;i++)                          (A + m.C + m.I)
{y = 1;                                     (A)
 a = CL[i][0];                              (S + 2.A)
 for (j=1;j <=a && y == 1;j++)              (A + 2.n'.C + n'.I)
       {b = x[ CL[i][j] ]                   (3.S + 3.A)
        y = y ∧ b;                          (C+A)
       }
 &b = &xf[ DV[i] ];                         (2.S + A)
 b = b ∨ y;                                 (C+A)
}
```

Computando todos os laços envolvidos, o tempo de execução do algoritmo será da forma:

$$T_t \leq A + 2.m.C + m.I + 6.m.A + 3.m.S + 3.m.n'.C + m.n'.I + m.n'.4.A + m.n'.3.S.$$

Supondo que A,C,I e S são funções lineares de um mesmo período T, tem-se que:

$$T_t \leq k1.T + k2.m.T + k3.m.n'.T,$$

ou seja, a ordem do algoritmo [1] é $O(m.n')$, onde m é o número de regras e n' é o número médio de antecedentes por regra.Como normalmente o valor de n' será sempre bem menor que o de m, pode-se considerar o algoritmo como de ordem $O(m)$.

## 6.4 Desempenho em Máquinas Paralelas

Em máquinas paralelas, o paralelismo induzido pela estrutura da matriz pode ser explorado, de forma a aumentar o desempenho..

Como uma máquina paralela não é convencional entre os mecanismos computacionais existentes, definir-se-á uma máquina paralela que atenda aos requisitos de maior tempo de processamento.

Em princípio, todas as informações que puderem ser processadas em paralelo o serão.

Na seção anterior, verificamos que basicamente o que era necessário mplementar era a operação $y = C \dot{\vee} x$ e em seguida efetuar $xf = x + D \dot{\wedge} y$.

Verifiquemos a implementação paralela de $y = C \dot{\vee} x$:

A mesma consta de se calcular, para todos os índices i e j a seguinte operação:

$$y_i = y_i \wedge (\overline{c}_{ij} \vee x_j)$$

Para se calcular esta matriz, é necessário portanto, que dois tipos de micro-máquinas sejam definidas: A primeira, deve tomar dois elementos e calcular o máximo dentre estes. A segunda, deve poder tomar n elementos, e calcular o mínimo dentre eles. Com estas duas micro-máquinas, todos os valores de i e j serão calculados ao mesmo tempo, e o tempo de execução da operação matricial como um todo será:

$$T_c = T_{\vee\_2} + T_{\wedge\_n}$$

onde $T_{\vee\_2}$ é o tempo de execução de uma máquina de máximos com 2 operandos e $T_{\wedge\_n}$ é o tempo de execução de uma máquina de mínimos com n operandos. Os tempos são somados, pois para que a máquina de mínimos seja executada, é necessário ter-se os resultados vindos da máquina de máximos.

O tempo de execução da operação disjuntiva ($D \dot{\wedge} y$) pode ser calculado de maneira semelhante, onde agora têm-se ao contrário a necessidade de máquinas de mínimos com dois operandos e máquinas de máximos com m operandos. De novo, a máquina com m operandos pode ser construída a partir de m-1 máquinas com apenas 2 operandos.

$$T_d = T_{\vee\_n} + T_{\wedge\_2}$$

Em seguida, é necessário ainda efetuar-se a soma lógica do resultado com o vetor de fatos original, o que pode ser efetuado por meio de mais uma série de máquinas de máximo de 2 operandos, levando a conta final do tempo de execução a:

$$T_t = 2.T_{v\_2} + T_{\wedge\_n} + T_{v\_n} + T_{\wedge\_2}$$

Supondo-se que estes tempos sejam todos iguais a T:

$$T_t = 5.T$$

O que faz com que o tempo de execução seja constante, em relação ao número de regras e ao número de antecedentes das regras, ou seja, sua ordem é $O(1)$.

Uma máquina paralela, como a descrita nesta seção, apesar de sua simplicidade conceitual, pode ser de difícil implementação, devido ao número de conexões que são necessárias para interligar as diversas micro-máquinas. Um esquema destas interconexões é mostrado na figura 6.1.

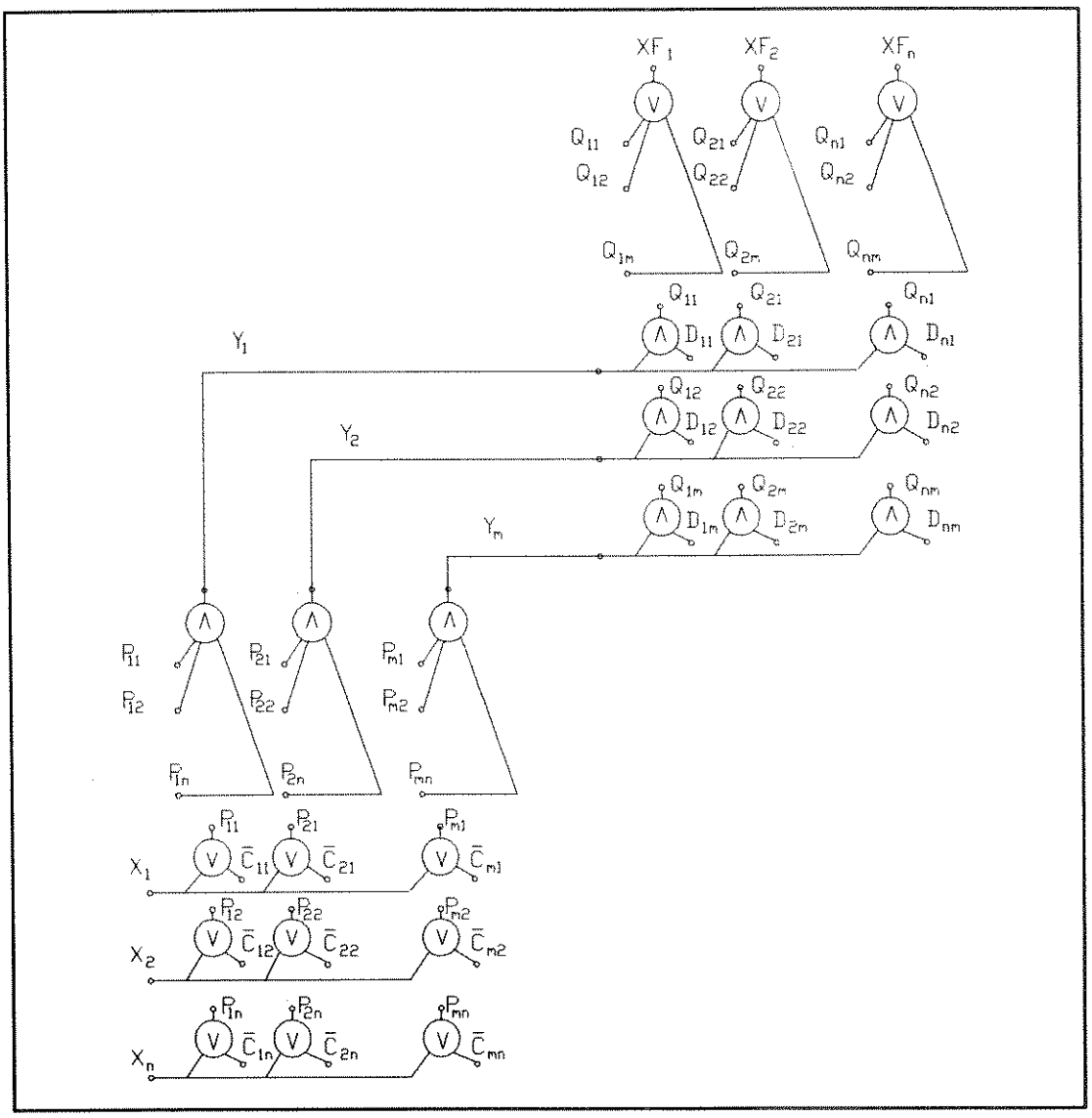


Figura 6.1 - Implementação de Máquina Paralela

A importância deste estudo, é que o mesmo possibilita a idealização de máquinas híbridas, ou seja, dotadas de certo paralelismo, mas não um paralelismo total. Estes tipos de máquinas, e a implementação do procedimento de inferência nos mesmos serão detalhados na seção a seguir, que trata do pseudo-paralelismo.

## 6.5 Paralelismo Parcial

Como foi visto na seção 6.4, a estrutura do procedimento permite uma grande paralelização em sua implementação. Entretanto, esta paralelização pode ser de difícil implementação, devido à necessidade do uso de máquinas especiais, e o grande número de interconexões entre elas. Isto não impede, todavia, que se use desta facilidade de paralelização, para melhorar o desempenho da implementação. Assim, pode-se aproveitar o caráter do procedimento, efetuando um paralelismo parcial, que hibridize a implementação sequencial e a paralela, aproveitando das vantagens de ambas. Caso se disponha de processadores em paralelo, pode-se ir alocando a cada processador, uma tarefa simples, referentes a índices i e j distintos. Essa não será entretanto, a abordagem utilizada nesta seção. Neste caso, supor-se-á que existe somente um processador, que opera instruções sequenciais. A implementação descrita aqui, será então um caso particular de paralelismo parcial, chamado de pseudo-paralelismo. Este nome vem do fato de a despeito de continuarmos a efetuar as operações sequencialmente, as mesmas poderão ser interpretadas como que conduzindo a um paralelismo parcial. Este tipo de implementação só será possível, entretanto, para casos que visem o processamento de regras proposicionais clássicas.

Suponha-se que as regras codificadas terão sempre um único consequente. Considere-se então que a matriz C é implementada por uma matriz do tipo $\alpha$-linha, e a matriz D seja reduzida a um vetor (devido ao número de consequentes), em uma implementação do tipo $\gamma$-coluna do tipo sem caracter delimitador. Neste caso, como tanto a matriz C como o vetor de fatos de entrada só possuem elementos binários, os mesmos podem ser guardados em bits, ao invés de bytes. Do mesmo modo, com elementos binários, as operações $\vee$ (max) e $\wedge$ (min) podem ser reduzidas a sua forma booleana, ou seja, as operações lógicas "ou" e "e". Dependendo dos processadores utilizados, os mesmos podem efetuar tais operações em 8, 16 ou 32 bits. Com isso, tais operações podem ser efetuadas de modo simultâneo, em pacotes de 8, 16 ou 32 operações por instrução de máquina. Esta técnica implementa portanto o pseudo-paralelismo, pois apesar de continuar a se trabalhar em máquinas sequenciais, as operações são executadas por meio de um paralelismo parcial.

Como exemplo de uma implementação deste tipo, suponha-se um processador de 16 bits, que opere com números inteiros de 2 bytes. Um possível algoritmo para implementar o pseudo-paralelismo será:

```
for(i=0;i <m;i++)
      {y = -1;
       for(j=0;(j <n_16) && (y == -1);j++)
                    y &= CLB [i][j]  | x[j];
       if(y == -1)
                    {a = 1;
                     b = DV [i] / 16;
                     c = DV [i] % 16;
                     xf[b] = (a << c) & xf[b];
                    }
      }
```

Neste algoritmo, m refere-se ao número de regras e n o número de símbolos utilizados na base de conhecimento. A matriz C está representada por meio de uma codificação do tipo $\alpha$-linha binária, ou seja, as colunas da matriz estão codificadas em pacotes de 16 bits, colocados em cada elemento da matriz de inteiros (2 bytes) *CLB*. Assim, quando se efetua a operação y &= *CLB*[i][j] | x[j], estão sendo efetuadas na verdade 16 operações, correspondendo as operações de :

$$y[i] = y[i] \wedge (c[i][16*j] \vee x[16*j])$$

a

$$y[i] = y[i] \wedge (c[i][16*j+15] \vee x[16*j+15])$$

Observe-se que o procedimento continua sendo exatamente o mesmo. Quando se coloca o valor de y em -1, o que se está fazendo é na verdade colocar todos os bits da variável y em 1, visto que -1 corresponde a 0xFFFF. Quando se verifica se a variável y continua em -1, isto significa que todas as operações $\wedge$ foram bem sucedidas, pois caso qualquer combinação ij resulte que $\bar{c}_{ij} \vee x_j = 0$, isto levará $y_i$ a 0, ou seja, y será diferente de -1. Caso y continue -1 após a operação, seta-se o bit correspondente ao consequente da regra em 1, indicando que a regra foi disparada. A vantagem deste algoritmo é que o loop interno do mesmo é efetuado apenas até n_16, que corresponde ao número de bytes necessários para acondicionar os n símbolos (n colunas da matriz C) em variáveis de 2 bytes. Assim, a ordem deste algoritmo é $O$(n_16.m). Comparado ao algoritmo sequencial que utiliza a implementação da matriz C como do tipo $\alpha$-linha simples, o algoritmo pseudo-paralelo é 16 vezes (ou 1600 %) mais rápido. Em máquinas de 32 bits que utilizem instruções de 32 bits, o mesmo será 32 vezes mais rápido.

Infelizmente o pseudo-paralelismo não pode ser utilizado quando se faz a inferência dentro do contexto da lógica nebulosa. Neste caso, é necessário definir-se um nível de quantização para a representação do fator de pertinência armazenado no vetor de fatos, o que torna o armazenamento em bits complicado, pois compromete a resolução do valor armazenado. O mais usual

é codificar-se este fator de pertinência em 1 ou 2 bytes, de modo que a estrutura do vetor de fatos fique de fácil acesso. Isto inviabiliza o pseudoparalelismo. Entretanto, quando se efetua um sistema híbrido, que contém tanto proposições nebulosas quanto proposições clássicas, pode-se também implementar uma máquina de inferência híbrida. Neste caso, deve-se codificar as proposições de modo que as colunas iniciais (ou finais) da matriz C sejam referentes a proposições nebulosas e as colunas finais (ou iniciais) sejam referentes às proposições clássicas. Havendo esta separabilidade, pode-se particionar a matriz, de modo que cada uma das sub-partições seja implementada por um tipo diferente de codificação. A sub-partição referente aos termos proposicionais clássicos pode ser implementada por uma sub-matriz do tipo $\alpha$-linha binária, o que faz com que a mesma possa ser tratada em termos de pseudo-paralelismo. Este tipo de matriz híbrida deve ser tratada por uma implementação também híbrida do procedimento. A utilização de esquemas híbridos pode ser fundamental na economia de espaço e de tempo, em alguns sistemas.

## 6.6 Comparações de Desempenho

Diversas comparações podem ser colocadas, no que tange ao desempenho do procedimento de inferência. As mais importantes são principalmente a da implementação em máquinas sequenciais e a implementação pseudo-paralela. Estas implementações podem ainda serem comparadas ao procedimento de encadeamento direto tradicional. Uma comparação da implementação em máquinas paralelas não pode ser efetuada pois não se dispõe, hoje, de dispositivos físicos que a efetivem. Entretanto, uma comparação pode ser efetuada diante da ordem do algoritmo utilizado.

O procedimento de inferência em encadeamento direto tradicional pode ser sumarizado no esquema 6.1, a seguir:

```
procedimento Encadeamento_Direto
{regras_disparadas = 0;
 do
      {disparo_no_passo = 0;
       PARA toda regra não marcada R[k]
       SE (todos os antecedentes de R[k] pertencem à BF)
                  {coloque o consequente de R[k] na BF;
                   marque a regra R[k];
                   regras_disparadas + +;
                   disparo_no_passo = 1;
                  }
      } while (regras_disparadas  N_REGRAS &&
                  disparo_no_passo = = 1)
}
```

**ESQUEMA 6.1 - Procedimento de Encadeamento Direto Tradicional**

Uma análise do procedimento de inferência via encadeamento direto tradicional, nos leva a constatar, que todo o conjunto de regras não disparadas (regra disparada = regra que tenha seus antecedentes satisfeitos e por isso tenha seu consequente adicionado à base de fatos) é processado continuamente, até que ou nenhuma regra mais dispare no ciclo, ou que todas as regras da base tenham sido disparadas. Portanto, o número de regras processadas será maior que o número de regras da base, visto que todas as regras não disparadas serão processadas novamente em passos ulteriores.

Neste particular, o procedimento de inferência via matrizes C-D já mostra claramente sua superioridade. Como o mesmo trabalha com bases de regras expandidas horizontalmente, não serão processadas, após o primeiro passo de inferência, todas as regras ainda não disparadas. O procedimento de expansão horizontal acrescenta para cada encadeamento formado na base de regras, uma nova regra que corresponde ao encadeamento. Assim, para cada encadeamento, será acrescentado apenas o processamento de uma nova regra, ao contrário do procedimento tradicional, quando todas as regras ainda não disparadas são reprocessadas. Esta diferença pode levar a uma economia de tempo de processamento vultuosa, quando se trabalha com grandes bases de regras, considerando-se que na maioria dos casos, não são todas as regras que disparam no primeiro passo, o que levaria, no procedimento tradicional, ao reprocessamento inútil repetidas vezes, de regras que não terão nenhuma influência no desenvolvimento do estado do sistema.

Outros tipos de comparações podem ser efetuados. Entretanto, os mesmo terão seu desempenho diretamente vinculados à implementação que se dá do procedimento tradicional. Como o procedimento é colocado, pode dar margem a uma miríade de interpretações, que gerarão diferentes tipos de implementações, algumas mais eficiente do que outras. De certa forma, o procedimento via matrizes C-D não deixa de ser um procedimento de encadeamento direto, visto sobre outro enfoque, que nos dá margens a otimizações.

Uma outra comparação importante diz respeito à implementação pseudo-paralela e a implementação sequencial do procedimento via matrizes C-D. Como se pode auferir das seções anteriores, a implementação pseudo-paralela possui um algoritmo de ordem $O$(n_16.m), ou seja, tem seu tempo de processamento diretamente proporcional ao produto de m (número de regras) e n_16 (número de símbolos utilizados em antecedentes dividido por 16). Já a implementação sequencial possui uma ordem $O$(n'.m), ou seja, seu tempo de processamento é proporcional ao produto de m e n' (média do número de antecedentes de uma regra). Comparando-se estas duas implementações, vemos que para as bases de regras, onde o número médio de antecedentes por regra seja menor do que o número total de antecedentes de toda a base dividido por 16, a implementação sequencial é mais eficiente, pois n' n_16. Entretanto, caso tenha-se n_16 n', a implementação pseudo-paralela acaba tornando-se mais eficiente, pois demandará menos cálculos. Em processadores de 32 bits, o mesmo acontece se n_32 n'. Com isso, diante de uma análise da base de regras, pode-se decidir qual a implementação mais eficiente dentre as apresentadas.

Uma implementação paralela, caso seja possível implementá-la efetivamente, será consideravelmente superior a todas as outras, visto que sua

complexidade é independente do número de regras, o que garante um tempo de processamento constante, independente da base de regras utilizada.

## 6.7 Resumo

Neste capítulo apresentou-se algumas possíveis implementações do procedimento de inferência, tanto em maquinas sequenciais, como em máquinas paralelas. Discutiu-se também o caso de implementações híbridas, como o pseudo-paralelismo. As implementações são então comparadas entre si e com o procedimento tradicional de encadeamento direto, quanto à eficiência de execução.

No capítulo seguinte será apresentada uma conclusão sobre o trabalho efetuado, bem como serão colocados pontos importantes na continuidade deste, como possíveis aperfeiçoamentos e extensões para instâncias superiores.

# 7. EXEMPLO DE APLICAÇÃO

## 7.1 Introdução

Neste capítulo será apresentado um exemplo da utilização do procedimento de inferência via matrizes C-D em uma aplicação bem determinada. Para tal, escolheu-se como aplicação o controle supervisório de um sistema de transporte vertical [4,5,36].

Nas seções seguintes, serão colocados os detalhes relevantes a respeito de sistemas transporte vertical, seguidos da descrição funcional da simulação utilizada. Em seguida serão discutidas algumas estratégias de alocação, incluindo-se uma estratégia de alocação nebulosa. Ao final, tem-se uma comparação dos resultados obtidos.

## 7.2 Sistema de Transporte Vertical

### 7.2.1 Grupo de Elevadores

Em prédios que possuem diversos pavimentos, é comum o uso de elevadores para proporcionar a locomoção de pessoas ou objetos entre os diferentes andares. O modo mais usual pelo qual estes elevadores são solicitados é pelo meio de botoeiras nos pavimentos e nas cabines, de modo que o usuário possa indicar para onde o elevador deve se dirigir para apanhá-lo, e em seguida para qual andar deve conduzí-lo [5].

Quando um único elevador mostra-se insuficiente para propiciar um transporte eficiente, dada a existência de um certo tráfego, a solução é utilizar mais de um elevador, formando-se então o que se chama de grupo de elevadores. Dentro de um grupo de elevadores, os mesmos podem funcionar independentes, ou seja, cada elevador apresenta botoeiras independentes de pavimento e de cabine, e cada carro é acionado em virtude do estado das diversas botoeiras de pavimento do prédio e de sua cabine. Entretanto, este não é o método mais eficiente, pois um usuário em um pavimento pode em princípio escolher um dos elevadores e acionar sua botoeira de pavimento, e este elevador não ser o mais adequado para atendê-lo, em virtude de estar lotado, ter várias paradas já programadas ou estar muito distante. Outra possibilidade seria o passageiro acionar as botoeiras de pavimento de todos os elevadores, fazendo com que várias paradas desnecessárias sejam programadas (pois apenas um elevador conduzirá o passageiro). Assim, o modo mais apropriado para se trabalhar com um grupo de elevadores, não é o serviço independente, mas o serviço em grupo.

Neste caso, um controlador supervisor administra todas as chamadas de pavimento e de cabine, alocando elevadores do grupo para atender as diversas chamadas, tentando otimizar estas alocações de modo que o transporte de passageiros no prédio seja o mais eficiente possível. Este controle é efetuado implementado-se o que se chama de estratégia de alocação, ou seja, a avaliação de uma demanda de chamadas, despachando convenientemente os elevadores de modo a cobrir esta demanda de modo eficiente.

### 7.2.2 Chamadas

As chamadas são registradas por meio das botoeiras que se encontram nos pavimentos e nas cabines. Quando funcionando em grupo, somente duas botoeiras são necessárias para cada pavimento, uma delas indicando uma chamada de subida e outra de descida. Já para cada cabine é necessária uma coleção de botoeiras com o mesmo número de andares acessíveis.

As chamadas de cabine e de pavimento diferem em alguns atributos.

Chamadas de pavimento são sempre chamadas com sentido, ou seja, o elevador deve se deslocar para o andar onde ocorreu a chamada, prevendo que em seguida será gerada uma chamada de cabine para um andar superior ou inferior ao atual, dependendo do sentido da chamada. Isso faz com que o elevador deve atender a uma chamada de subida quando está subindo e a uma chamada de descida quando está descendo, a menos que mude de sentido para atender a uma determinada chamada. A opção de mudar ou não de sentido em uma determinada situação será decidida pela estratégia de alocação.

Já as chamadas de cabine são sempre chamadas sem sentido, ou seja, o elevador deve se dirigir para o andar da chamada, sem compromisso se a chamada seguinte fará o elevador subir ou descer. Tais chamadas devem portanto ser atendidas o mais rápido possível, seja quando o elevador estiver com sentido de subida como de descida.

Outro atributo das chamadas é em relação à previsão do fluxo de passageiros. Uma chamada de pavimento, seja de subida ou de descida, prevê sempre a entrada de um passageiro na cabine. Ao contrário, uma chamada de cabine prevê sempre a descida de um passageiro. Estes atributos das chamadas são utilizados durante a definição de uma estratégia de alocação, de modo que a mesma possa prover a eficiência que se espera do sistema.

### 7.2.3 Controlador do Carro e Vetor de Alocações

O controlador do carro é um sistema de controle individual de cada elevador. Ou seja, cada controlador de carro atua somente em um elevador, e sua função é levar o elevador do andar atual até um novo andar, propiciando a aceleração e desaceleração do mesmo, bem como o abrir e fechar de portas, e o tempo em que a porta permanece aberta para a entrada e saída de passageiros.

Apesar de ser um controle independente por si, o controlador do carro manipula parâmetros que são importantes no planejamento da estratégia de alocação, tais como a posição, velocidade e aceleração do carro, o estado das portas (aberta / abrindo / fechando / fechada), *timers* e controladores de entrada

de passageiros (foto-célula). Todos estes parâmetros,além do estado das botoeiras, devem estar disponíveis de modo que o controlador de grupo possa tomar as decisões que implicarão na alocação dos elevadores.

Do mesmo modo, o controlador de grupo deverá administrar um vetor de alocações para cada elevador do grupo, onde deverá indicar a próxima parada (parâmetro que será enviado aos controladores de carro) e todas as demais paradas previstas alocadas ao carro, de modo que o mecanismo de controle supervisório possa definir as alocações das novas chamadas que vão aparecendo.

### 7.2.4 Tráfego

Outro elemento importante que deve ser analisado pelo controlador de grupo diz respeito às condições de tráfego, que definirão as tendências de fluxo de passageiros na dinâmica do prédio.

Alguns padrões de tráfego podem ser analisados e, eventualmente, participarem da estratégia de alocação a ser implementada, dada sua ocorrência mais acentuada de forma geral. Padrões particulares de tráfego poderão ser incorporados em uma estratégia de alocação, desde que se tenham mecanismo pelos quais os mesmo possam ser descritos.

Os padrões mais comuns de tráfego são os seguintes:

- Serviço Leve
- Tráfego Inter-Andares
- Pico de Subida
- Pico de Descida

O serviço leve é um padrão de tráfego caracterizado pela baixa taxa de chamadas em todos os andares, não exigindo muito do sistema. Este padrão ocorre geralmente em determinadas horas do dia, como por exemplo durante a madrugada, quando o sistema de elevadores é pouco solicitado.

O tráfego inter-andares, corresponde a uma condição onde existem chamadas esporádicas de andares para andares, ocorrendo geralmente durante o expediente, em prédios comerciais, quando existem passageiros trafegando entre os andares, a uma taxa mais ou menos uniforme.

O pico de subida é um padrão muito bem determinado, correspondente a um grande número de chamadas do pavimento principal do prédio para os outros andares. Geralmente ocorre no início do dia, em prédios comerciais ou ao final do dia em prédios residenciais. É um padrão que normalmente necessita de uma estratégia especial, visto que a demanda por elevadores no pavimento principal pode crescer muito, prejudicando a eficiência do transporte.

O pico de descida corresponde ao contrário do anterior, quando existe uma forte demanda de todos os andares, de chamadas em direção ao pavimento principal. Também é uma situação razoavelmente crítica do sistema, onde estratégias especiais devem ser adotadas de modo a considerar características do padrão de chamadas.

Normalmente o tráfego em um prédio passa por todos estes padrões durante o transcorrer do dia, podendo inclusive alguns deles (como por exemplo tráfego inter-andares e pico de descida, etc) coexistir. Uma boa estratégia de

alocação deve considerar a existência de padrões de tráfego e tomar decisões que propiciem um transporte adequado.

### 7.2.5 Critérios de Desempenho

De um modo geral podem ser colocados diversos critérios de desempenho, a fim de que se possa classificar o grau de atendimento de uma determinada estratégia de alocação. Estes critérios são utilizados então, para que se possa definir a melhor estratégia de alocação para uma dada situação. Exemplos de critérios seriam:

- tempo de espera
- tempo de percurso
- tempo total de atendimento
- consumo de energia
- atendimento preferencial

O critério do tempo de espera busca minimizar o tempo de espera médio, ou seja, o tempo transcorrido desde que o passageiro aperta a botoeira de pavimento até o momento em que o mesmo entra no elevador.

O critério do tempo de percurso espera minimizar o tempo médio de percurso, isto é, o tempo transcorrido desde que o passageiro entra no elevador e aperta a botoeira de cabine, até que o mesmo desça no andar desejado.

Um critério que conjuga os dois critérios anteriores é o critério do tempo total de atendimento, o qual busca minimizar o tempo de atendimento médio, que é o tempo transcorrido desde que o passageiro aperta a botoeira de cabine até que o mesmo desça no andar desejado.

Uma outra estratégia pode tentar minimizar o consumo de energia por parte dos elevadores.

Além destas, pode-se definir critérios em que a meta seja dar um atendimento preferencial a determinados andares, tais como atendimento VIP, andares onde existam restaurantes e cinemas, etc, que devem ter, em certas situações,preferência sobre as demais chamadas.

Em geral, o que se objetiva em uma estratégia de alocação, é encontrar uma solução onde estes critérios sejam envolvidos e, dependendo do tráfego, seja dada relevância maior a um ou outro critério. Uma boa estratégia de alocação deve considerar o maior leque de alternativas possível.

## 7.3 Simulação

Para testar uma estratégia de alocação, recorre-se a uma simulação do sistema de elevadores. Para nossos propósitos, procedeu-se a uma simulação de um grupo de elevadores de modo a testar o controlador de grupo e, de certa forma, validar o procedimento de inferência via matrizes C-D como uma

alternativa utilizada em uma aplicação real, de implementação simples e factível.

O simulador de grupo de elevadores é constituído de três módulos básicos, correspondentes a:

- Simulação de Chamadas
- Simulação da Dinâmica dos Elevadores
- Controlador de Grupo

A simulação de chamadas é o módulo responsável pela definição do tráfego que se deseja simular no prédio, e seu acondicionamento em tabelas que serão utilizadas pelo simulador da dinâmica dos elevadores.

A simulação da dinâmica dos elevadores, considera as chamadas geradas pelo módulo de geração de chamadas, além das decisões colocadas pelo controlador de grupo, e simula a movimentação dos elevadores pelos andares do prédio, além de considerar todos os detalhes do controlador de carro, tais como sentidos dos elevadores, abertura e fechamento das portas, tempo de não-interferência, tempo de fotocélula, etc

O módulo controlador de grupo administra a fila de alocações dos elevadores e, a partir dos dados de estado dos elevadores e do estado das botoeiras de pavimento e de cabine, atualiza e gerencia esta fila, enviando os próximos andares-meta para cada um dos elevadores, de acordo com as estratégias de alocação utilizadas.

### 7.3.1 Simulação de Chamadas

A simulação de chamadas é efetuada baseada na descrição do tráfego desejado. Basicamente considera-se as situações de tráfego inter-andares, pico de descida e pico de subida. As chamadas são geradas por meio de dois geradores de números pseudo-aleatórios, o primeiro fornecendo uma distribuição uniforme e o segundo uma distribuição gaussiana. Estas são simuladas atribuindo-se o horário de sua ocorrência, sua origem e seu destino.

Na simulação de chamadas inter-andares utiliza-se o gerador de números com distribuição uniforme, que fornecede os dados que irão determinar a hora, a origem e o destino das chamadas.

Para o pico de subida, o horário da chamada é determinado pelo gerador de números com distribuição do tipo gaussiana, o que faz concentrar um grande número de chamadas em um determinado intervalo de tempo. A origem é sempre o pavimento principal e o destino é dado pelo gerador de números de distribuição uniforme, o que garante a mesma probabilidade de existência de chamada para cada andar. Para o pico de descida, o horário da chamada também é oriundo do gerador de números de distribuição gaussiana mas, ao contrário do pico de subida, a origem é fornecida pelo gerador de números com distribuição uniforme e o destino é sempre o pavimento principal.

A determinação de uma sequência de simulação permite a mistura dos padrões de picos com o de chamadas inter-andares. Deste modo o usuário fornece o número de chamadas desejadas em pico (subida ou descida) e o número de chamadas desejadas inter-andares. As chamadas são então

acondicionadas em tabelas que são posteriormente acessadas pelo simulador da dinâmica de elevadores.

### 7.3.2 Simulação da Dinâmica dos Elevadores

A simulação da dinâmica dos elevadores segue a seguinte sequência:

```
INICIALIZAÇÃO
ENQUANTO (RELÓGIO != PERÍODO DE SIMULAÇÃO)
{ PROCESSA CHAMADAS
 ALOCA CHAMADAS DE PAVIMENTO
 PROCESSA STATUS DOS MOTORES
 MOVIMENTA ELEVADORES
 VERIFICA META
 EXIBE SINÓTICO
 ATUALIZA RELÓGIO
}
EXIBE RELATÓRIOS
```

Na inicialização é feita a geração das chamadas, e os diversos vetores e variáveis utilizados são inicializados.

Um relógio é atualizado a cada ciclo, de modo a marcar o instante de ocorrência de cada evento, e o estado associado.

O ciclo de simulação começa com o processamento de chamadas. Neste, o relógio é comparado com a tabela de chamadas e caso verifique-se uma chamada no instante, a mesma é colocada na fila de pavimento, simulando-se uma botoeira de chamada de pavimento sendo apertada. Em seguida as chamadas são analisadas pelo controlador de grupo e alocadas internamente.

A etapa seguinte processa o estado dos motores, ligando-se ou desligando-se os mesmos e determinando-se seu sentido, em virtude do próximo andar determinado pelo controlador de grupo. Baseada no estado dos motores é feita então a movimentação dos elevadores, no sentido correto, quando a mesma existir. Na sequência, verica-se se a meta foi atingida, ou seja, se o andar corrente é o mesmo determinado como próximo andar pelo controlador de grupo. Neste caso, simula-se a parada do elevador, a saída eventual de passageiros do carro, a entrada eventual de novos passageiros, e o aperto das botoeiras de cabine pelos passageiros ingressantes.

Exibe-se então o sinótico (o novo estado do sistema, representado de maneira pictórica), e se atualiza o relógio e os *timers*.

Ao terminar o ciclo de simulação, emitem-se os relatórios de tempo de espera de cada chamada e do tempo de espera médio.

### 7.3.3 Controlador de Grupo

Na simulação, implementou-se três tipos de controladores de grupo, todos eles manuseando um vetor de alocações para cada elevador, onde são

inseridas e suprimidas chamadas, em diferentes posições. As estratégias de alocação utilizadas serão discutidas na seção 7.4.

Os vetores de alocação são basicamente filas, que tem uma dimensão controlada e que podem ter elementos inseridos em qualquer ponto da mesma. Supondo uma representação em que o primeiro elemento da fila esteja à esquerda e o último a direita, a inserção de um elemento (chamada) em um ponto da fila, faz com que os elementos a sua direita desloquem-se de uma posição a direita, incrementando-se a dimensão da fila. Do mesmo modo, a supressão de um elemento da fila faz com que os elementos a sua direita desloquem-se de uma posição a esquerda, decrementando-se a dimensão da fila.

A fila pode em qualquer momento ser particionada em três segmentos, podendo qualquer um deles estar vazio, exceto o 1º, quando o 2º ou o 3º contém elementos. Cada segmento orientará a viagem do elevador em um sentido. A inserção de uma chamada na fila será realizada no segmento adequado a conter a chamada, de modo que a mesma fique ordenada dentro do segmento. Uma chamada de pavimento de subida deve necessariamente ser inserida em um segmento de subida. Do mesmo modo uma chamada de pavimento de descida deve ser inserida em um segmento de descida. A ordem em que se encontram os segmentos será dada pelo primeiro elemento do vetor de alocações, diante do andar corrente do elevador. Se a próxima chamada for acima do andar atual, então o 1º e o 3º segmentos são de subida, sendo o 1º responsável por todos os andares acima do andar atual e o 3º por todos os andares abaixo do atual (com chamadas em sentido de subida). Neste caso o 2º segmento será de descida. Para o caso da próxima chamada ser abaixo do andar atual, o raciocínio é o contrário. As chamadas de cabine podem ser colocadas em princípio em qualquer um dos segmentos. Serão colocadas no 1º segmento, se puderem ser atendidas no sentido atual do elevador. Caso contrário será colocadas no 2º segmento. Chamadas de cabine nunca serão colocadas no 3º segmento.

Esta segmentação do vetor de alocação é utilizada para se determinar, a partir de um vetor preenchido, qual deve ser o ponto de inserção de uma nova chamada, de modo que se preserve o sentido de movimentação dos elevadores. Diz-se então que a chamada é inserida no lugar apropriado, diante da administração que é feita pelo controlador de grupo.

A medida que as chamadas vão sendo atendidas, o controlador de grupo vai suprimindo-as do vetor de alocação, deslocando-se o vetor para a esquerda.

## 7.4 Estratégias de Alocação

Basicamente foram simuladas 3 estratégias de alocação. As duas primeiras foram colocadas mais a nível de referência, sendo que a terceira é realizada efetivamente utilizando-se o procedimento de inferência via matrizes C-D nebulosas, onde adotou-se 9 regras para efetuar a alocação.

A primeira estratégia é simplesmente uma estratégia de referência e consiste basicamente de escolher um elevador ao acaso e alocá-lo para atender a chamada. Esta estratégia obviamente não atende a nenhum critério de

desempemho, sendo colocada somente para se comparar os resultados entre uma estratégia nula e uma estratégia que efetivamente se norteie por um critério de desempenho determinado.

A segunda estratégia, apesar de não ser uma estratégia colocada a base de regras, objetiva atender a um critério de desempenho, que é o de minimizar o tempo de espera médio. Sua ação consiste em, para cada chamada que ocorre, calcular-se o tempo de atendimento previsto para cada elevador, ou seja, o tempo que cada elevador, diante de sua posição e das chamadas que já estão a ele alocadas, demorará para atender a chamada. Em seguida, aloca-se a chamada ao elevador que tiver o menor tempo de atendimento previsto. Esta estratégia claramente visa minimizar o tempo de espera médio, visto que o parâmetro calculado como tempo de atendimento, tende a ser uma primeira avaliação do tempo de espera da chamada (pois podem ser acrescentadas mais chamadas posteriormente que atrasem o tempo de atendimento calculado a princípio).

A terceira estratégia consiste de uma base de regras nebulosa a qual define um parâmetro chamado prioridade, utilizado para decidir a alocação.

As regras são regras nebulosas do seguinte tipo:

SE te é BAIXO E ta é BAIXO ENTÃO pr é MÉDIA.
SE te é MÉDIO E ta é BAIXO ENTÃO pr é ALTO.
...

Nestas regras, te é o tempo de espera, e é calculado como sendo o tempo que uma chamada C já está esperando, ou seja, o tempo transcorrido desde que a botoeira de pavimento foi apertada até o instante da decisão, ou seja o instante atual. Já ta é chamado de tempo de atendimento, e é calculado como uma previsão que se tem do tempo necessário para se atender a chamada. Este parâmetro é calculado verificando-se a posição de inserção da chamada no vetor de alocação do elevador, e calculando-se o tempo de percurso do elevador até chegar na posição determinada, acrescentado do tempo de abertura, entrada/saída de passageiros e fechamento de portas, para cada chamada que antecede a calculada. O parâmetro prioridade é um parâmetro relativo, colocado entre 0 e 100, que irá definir a prioridade que a chamada tem de ser atendida pelo elevador em questão.

Na estratégia nebulosa, adotou-se que somente uma chamada de pavimento poderá ser alocada a cada elevador, permanecendo todas as outras como pendentes. Este esquema permite a realocação de chamadas, pois a cada ciclo de decisão, todas as alocações são reavaliadas. As prioridades são calculadas a partir do conjunto de regras, sendo que para cada elevador é montada uma fila de prioridades de atendimento, baseada nas prioridades de atendimento a cada chamada. A partir destas filas, é feita uma disputa por chamadas, onde cada elevador se candidata para atender a chamada de maior prioridade de sua fila. Em caso de conflito, ou seja, de mais de um elevador desejar atender a mesma chamada, o critério de desempate passa a ser o tempo de atendimento da chamada, ganhando o elevador que possuir o menor tempo de atendimento, passando o elevador perdedor a disputar a sua segunda

prioridade, e assim por diante, até que cada elevador fique com uma chamada distinta, que é então efetivamente alocada.

No ciclo seguinte, as chamadas de pavimento corrente são re-analisadas. As chamadas que estão alocadas em primeira instância, ou seja, são as chamadas às quais os elevadores estão se dirigindo imediatamente para atender, não são alteradas. Já as chamadas que estão alocadas em segunda instância, ou seja, não são as chamadas que estão na iminência de serem atendidas, são desalocadas, passando a competir de novo com as demais chamadas que ainda não foram alocadas.

Esta sucessão de ciclos de inferência ocorre durante toda a simulação, agindo realmente como um supervisor de grupo, orientando a cada instante como deve ser a alocação, de modo a atender a demanda, de acordo com o critério de desempenho escolhido que é o de se tentar minimizar o tempo de espera médio das chamadas.

## 7.5 Inferência Nebulosa

Como foi visto no capítulo 5, a inferência nebulosa como um processo completo está dividida entre as fases de pré-processamento, inferência e pós-processamento. Nesta seção são descritas, para o caso do controle de grupo de elevadores, as etapas desenvolvidas por cada fase.

### 7.5.1 Pré-Processamento

Na fase do pré-processamento, basicamente o que se faz é gerar os vetores de fatos de entrada, a partir dos dados colhidos do meio. Segue-se o seguinte roteiro:

- Preencher o vetor de chamadas pendentes
- Calcular te para cada chamada pendente
- Calcular ta para cada chamada pendente
- Fuzzyficar te's e ta's
- Preencher o vetor de fatos nebulosos

Inicialmente, verifica-se as chamadas de pavimento já alocadas e aquelas que não estão alocadas em primeira instância e desalocam-se as mesmas. Em seguida toma-se um vetor de chamadas pendentes vazio, e passa-se a preenchê-lo com as chamadas de pavimento existentes. Ao final desta etapa, o vetor de chamadas pendentes conterá todas as chamadas de pavimento existentes que não estão alocadas em primeira instância, ou seja, as chamadas que disputarão pela alocação.

O passo seguinte é calcular, para cada chamada, os tempos de espera e os tempos de atendimento diante de cada elevador.

Em seguida fuzzyfica-se o os tempos de espera e tempos de atendimento, ou seja, toma-se para cada instância da variável linguística em

questão o valor da função de pertinência do mesmo no ponto onde ocorre a amostragem, ou seja, o tempo de espera e tempo de atendimento calculados. Com estes valores são gerados os vetores de fatos de entrada (um para cada chamada/elevador). Neste ponto, o processo encontra-se pronto para a fase da inferência.

### 7.5.2 Inferência

A fase da inferência é realizada pelo procedimento de inferência via matrizes C-D, sendo realizada um ciclo de inferência completo para cada par chamada/elevador. O procedimento de inferência irá gerar então os vetores de fatos nebulosos inferidos relativos às instâncias linguísticas da variável linguística prioridade (também para cada par chamada/elevador), que poderá então passar ao pós-processamento para finalizar a inferência nebulosa completa.

### 7.5.3 Pós-Processamento

Na fase de pós-processamento implementa-se os seguintes passos:

- Defuzzyfica os vetores de fatos inferidos de prioridades.
- Efetua a disputa por chamadas.
- Obtém a decisão final das alocações dos elevadores.

A etapa da defuzzyficação, toma para cada par chamada/elevador, o vetor de fatos inferidos de prioridade, efetua a agregação linguística dos termos e defuzzyfica a prioridade nebulosa, gerando um valor de prioridade, obtido pelo método do centro de massa, que é o mais representativo diante da função de pertinência inferida.

Estas prioridades são então ordenadas por elevador, gerando as filas de prioridade para cada elevador, que irão alimentar a disputa por chamadas.

A disputa por chamadas decidirá então, para cada elevador, qual a chamada que deve a ele ser alocada. Esta decisão será então inserida nos vetores de alocação de cada elevador, finalizando a inferência nebulosa e o controle nebuloso em si.

## 7.6 Resultados

Foram feitas simulações de várias condições de tráfego. Estas simulações foram efetuadas tendo-se como parâmetros os seguintes:

- Nº de Andares : 7
- Nº de Elevadores : 4
- Capacidade dos elevadores : 6 pass.
- Velocidade dos Elevadores : 3 m/s
- Patamar por andar : 3 m
- Tempo de Abertura de Porta : 2 s
- Tempo de Não-Interferência : 2 s
- Tempo de Fechamento de Porta : 2 s
- Período de Simulação (Tique do relógio) : 500 ms
- Tempo Total de Simulação : 550 s (9 min e 10 s)
- Condições de Tráfego : até 34 pass/min (no pico).
- Padrões de Tráfego : Tráfego Inter-andares, pico de subida e pico de descida

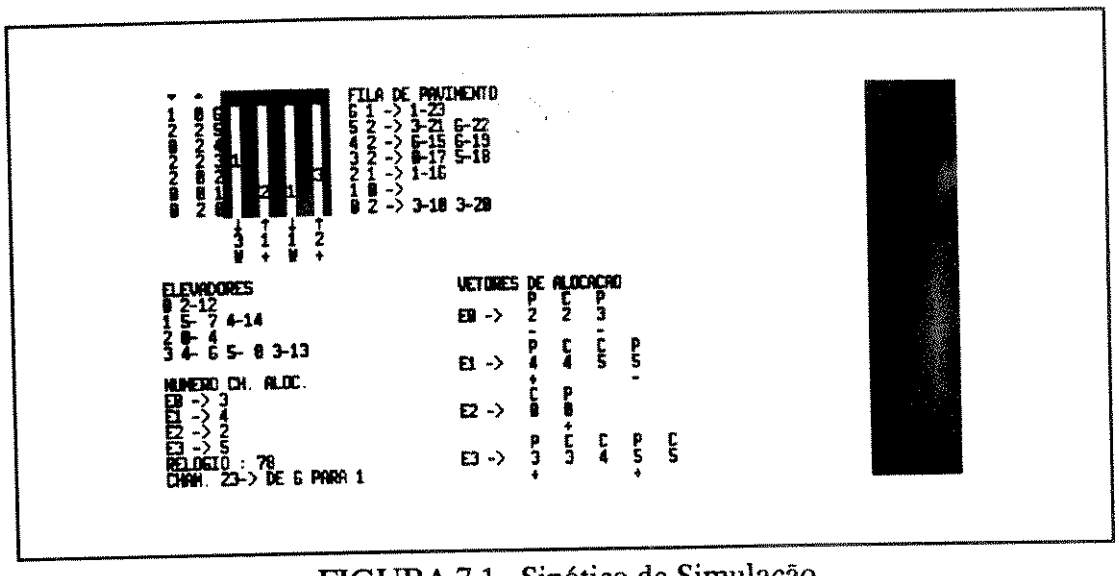


FIGURA 7.1 - Sinótico de Simulação

As simulações foram feitas apresentando-se a mesma característica de tráfego, e simulando-se diversas estratégias de alocação diante desta, incluindo-se as estratégias não-nebulosas (randômica e mínimo tempo de atendimento), além de diversas estratégias nebulosas, onde a diferença entre estas consiste em uma escolha de diferentes funções de pertinência, tendo como diferença básica a escolha dos universos de discurso para os diferentes parâmetros.

Na figura 7.1, pode-se observar um exemplo de sinótico de simulação. Nas figuras 7.2 e 7.3 tem-se o resultado de uma simulação de um padrão de tráfego uniforme, com 20 passageiros, para o controlador não-nebuloso de mínimo tempo de atendimento. Nas figuras 7.4 e 7.5, tem-se a mesma situação simulada com um controlador nebuloso.

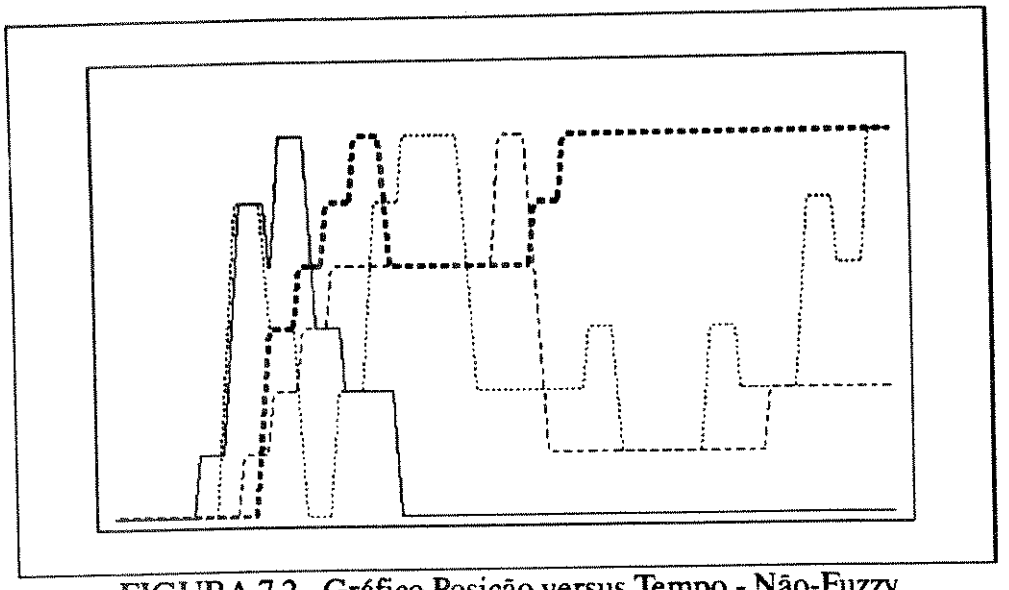

FIGURA 7.2 - Gráfico Posição versus Tempo - Não-Fuzzy

As características de tráfego foram definidos baseadas nos padrões de tráfego, que são o tráfego inter-andares, o pico de subida e o pico de descida. Diante do tempo de simulação constante, variou-se o número de chamadas em cada tipo de padrão de tráfego. O tráfego mais intenso (100 chamadas em pico de subida) corresponde a aproximadamente 34 pass/min. Nestas simulações adotou-se combinações de padrões de tráfego de modo a gerar resultados mais realistas. No total, nenhuma combinação excede às 100 chamadas. Para cada característica de tráfego, mediu-se o tempo de espera médio das chamadas.

A partir das simulações pode-se tecer uma série de considerações a respeito da estratégia de alocação utilizada, bem como apontar falhas em seu comportamento. Estas conclusões são baseadas nas coletas dos tempos de espera médio para cada característica de tráfego, além da própria observação do sinótico de simulação.

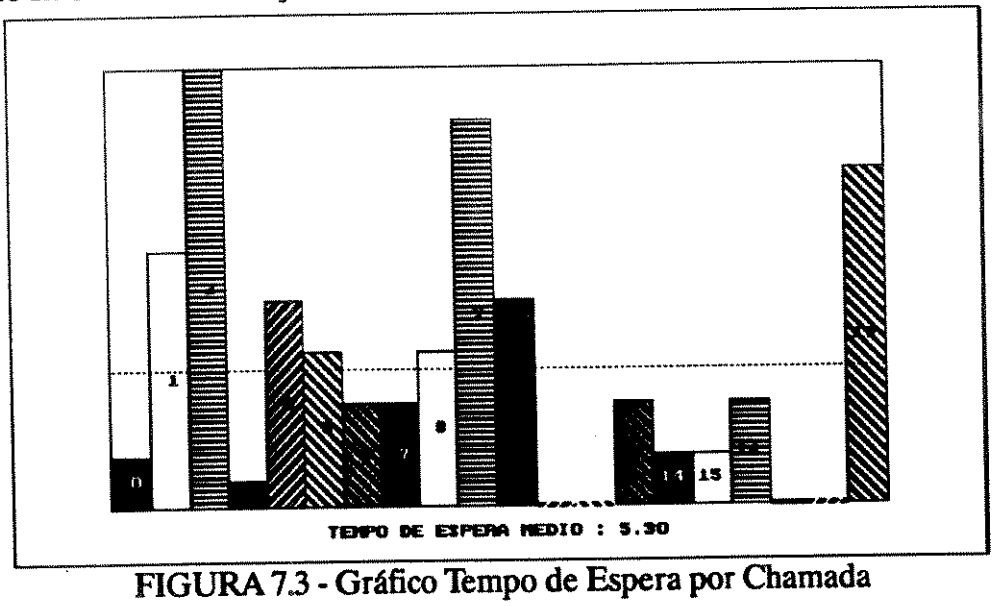


FIGURA 7.3 - Gráfico Tempo de Espera por Chamada

Uma das primeiras observações que ocorrem é que a estratégia nebulosa que permite realocações é superior, pelo menos na maioria dos casos, apresentando um tempo de espera médio que é 14 % em média menor que a estratégia sem realocações do mínimo tempo de atendimento.

Uma observação mais sutil é que nem sempre a estratégia com realocação é melhor. Em diversos casos, notou-se que a estratégia sem realocação obtinha um melhor tempo de espera médio para uma dada situação. Esta diferença não é oriunda somente da falta de uma sintonia fina das funções de pertinência (o que poderia explicar tal comportamento a princípio), mas de uma situação em que, efetuando-se a realocação, ou seja, realocando-se as chamadas de modo que os melhores elevadores atendam às chamadas mais prementes, algumas vezes processa-se um desbalanceamento do prédio, isto é, colocam-se a maioria dos elevadores em uma dada região do mesmo, fazendo com que chamadas em outras regiões sejam prejudicadas, tendo um tempo de atendimento alto para todos os elevadores. Nesta situação tenta-se otimizar o atendimento mas acaba-se por prejudicá-lo, pois não se leva em consideração

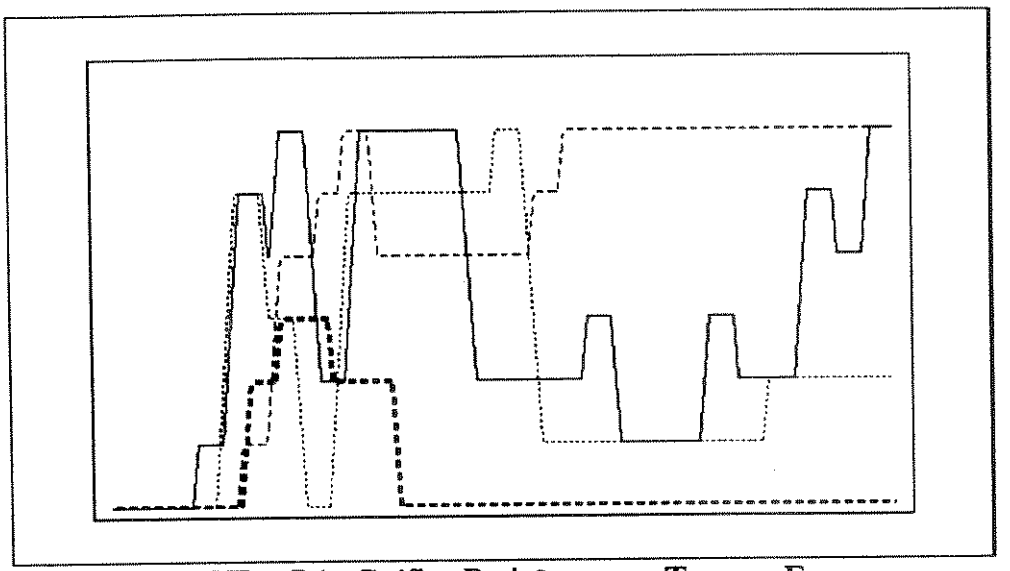

FIGURA 7.4 - Gráfico Posição versus Tempo - Fuzzy

as chamadas futuras, ou seja, uma previsão a respeito das demais chamadas, de modo a evitar desbalanceamentos deste tipo.

Outra falha que a estratégia apresenta, na forma atual, é o fato de não considerar a ocupação interna dos elevadores como fator de decisão das alocações. Isto faz com que, durante tráfego intenso, um elevador lotado pare para atender uma chamada de pavimento que não tem condição de ser atendida pelo mesmo, visto que o mesmo não comporta mais passageiros.

Do mesmo modo, a estratégia não privilegia chamadas onde existe uma grande fila de passageiros (pois para cada pavimento existe apenas uma botoeira, que não dimensiona em princípio o tamanho da demanda desta chamada). Para estes casos, poderia-se sugerir uma estratégia que incorporasse uma alocação múltipla, onde mais de um carro seriam enviados para atender a uma mesma chamada, desde que a demanda desta chamada fosse considerada alta, por exemplo.

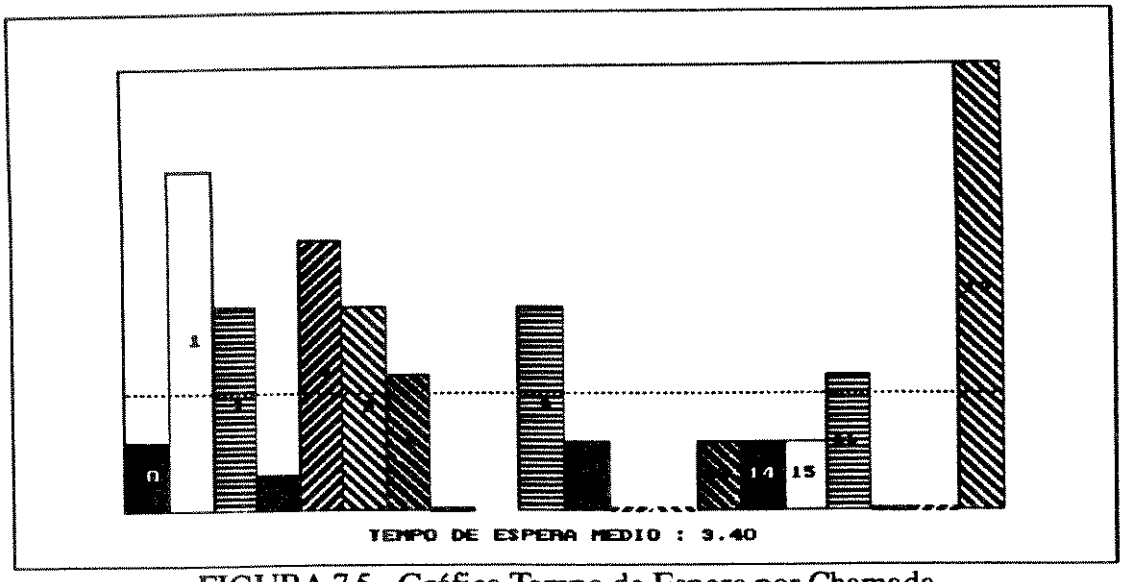


FIGURA 7.5 - Gráfico Tempo de Espera por Chamada

Estas lacunas na estratégia, fazem com que alguns padrões de tráfego tenham seu desempenho prejudicado. Por exemplo, o fato de não se privilegiar o tamanho da fila de passageiros, faz com que não se atenda a contento situações de pico de subida, por exemplo, onde a demanda por transporte no pavimento principal é diferenciadamente maior que nos demais pavimentos. Nestes casos, uma chamada com um único passageiro será tratada do mesmo modo que uma chamada no pavimento principal, com um grande número de passageiros esperando. A impossibilidade de se fazer alocações múltiplas também não contribui para o desempenho da estratégia.

Para o caso de pico de descida, o fato de não se considerar o estado de ocupação da cabine, faz com que os andares mais baixos sejam prejudicados, pois os elevadores têm a tendência de subirem para atender primeiro as chamadas de descida mais altas, e quando chegam nos andares mais baixos, já estão lotados, fazendo com que ocorram congestionamentos nos andares mais baixos.

O fato de não se utilizar presentemente um esquema preditivo, prejudica a estratégia em todos os tipos de padrões, fazendo com que escolhas aparentemente mais efeicientes de chamadas, levem a desbalanceamento do atendimento, o que pode comprometer o desempenho.

É certo que a maioria destas faltas não ocorre somente na alocação nebulosa com realocação. Entretanto, tendo-se em conta que uma arquitetura baseada em lógica nebulosa permite o tratamento de tais ítens, o levantamento destes constitue uma importante diretriz na escolha de uma estratégia mais abrangente, baseada nos mesmos princípios.

**Deste modo conclui-se que o uso de lógica nebulosa, mais especificamente do procedimento de inferência via matrizes C-D nebulosas, pode ser um paradigma importante na implementação de sistemas de controle supervisório. A facilidade com que se desenvolve modelos de controle, e consideram-se múltiplos critérios, faz com que a lógica nebulosa torne-se de grande valia na implementação de tais sistemas, mesmo em tempo real,**

propiciando ao mesmo tempo uma depuração no algoritmo de controle de modo bem simples.

## 7.7 Resumo

Neste capítulo, efetuou-se a implementação de um exemplo de aplicação, onde foi utilizado o procedimento de inferência via matrizes C-D nebulosas.

Para ilustrar uma aplicação modelou-se um sistema de transporte vertical (controle de grupo de elevadores), implementando-se um simulador para o mesmo, sendo que o controle do sistema foi projetado utilizando-se lógica nebulosa, representada pelo procedimento de inferência via matrizes C-D nebulosas.

O exemplo serviu para mostrar a factibilidade do uso do procedimento bem como sua aplicabilidade a problemas do mundo real.

No capítulo seguinte são colocadas as conclusões gerais sobre o procedimento de inferência, além de sugestões de trabalhos futuros.

# 8. CONCLUSÕES E TRABALHOS FUTUROS

Após a definição e análise do procedimento de inferência via matrizes C-D, seguidas de sua utilização em um exemplo de aplicação, pode-se concluir que o mesmo atingiu as expectativas que motivaram o esforço em seu desenvolvimento.

Uma análise dos requisitos para sistemas de processamento de conhecimento em tempo real (apresentados na seção 1.4), permite dizer que o procedimento de inferência via matrizes C-D atende à maioria dos mesmos.

Quanto à integração eficiente do processamento numérico-simbólico, a mesma depende diretamente do processamento auxiliar (pré-processamento e pós-processamento) necessário para a implementação do procedimento. Para tal, o procedimento assegura grande liberdade, permitindo ao projetista atingir tal requisito.

Quanto ao requisito da existência de mecanismos de foco de atenção, o procedimento é bastante flexível. Como a base de regras fica praticamente limitada a uma matriz, a mesma pode ser facilmente chaveada durante a execução do programa, implementando uma base de regras estruturada, onde cada base (ou sub-base) será considerada por uma matriz. Do mesmo modo, o chaveamento de uma matriz é feito facilmente, o que possibilita uma implementação eficiente que atende este requisito.

O procedimento também administra de modo eficaz o tempo de execução, desde que não considera módulos explicativos, e praticamente elimina o encadeamento de regras.

Além disso, o procedimento via matrizes C-D pode sempre garantir um tempo de resposta, devido ao seu caráter matricial, o que possibilita sua aplicação em sistemas sob restrições severas de tempo.

Outro requisito satisfeito é aquele que diz respeito à não monotonicidade de alguns sistemas controlados. Como o procedimento de inferência é muito rápido, o mesmo pode ser invocado periodicamente, com intervalos entre atuações bem pequenos, de modo que em tais intervalos, o sistema pode ser considerado monotônico. Com isso, consegue-se uma aproximação ao controle de sistemas não-monotônicos, visto que os dados são constantemente atualizados do processo. No entanto, do ponto de vista estrito esta questão permanece em aberto.

Outra característica é que, devido ao seu caráter matricial, o mesmo pode ser implementado facilmente em qualquer tipo de máquina. Devido ao mesmo motivo, os mesmos são de fácil integração com outros tipos de software.

Diante destas constatações, verifica-se que de fato o procedimento atingiu seus objetivos básicos.

Entretanto, pode-se detectar uma série de aperfeiçoamentos que podem ser colocados, de modo a dar continuidade e aperfeiçoar este trabalho.

Em primeiro lugar, a lógica proposicional, mesmo com sua generalização na lógica nebulosa, nem sempre é suficiente para a modelagem adequada de classes de sistemas mais complexos. Quando a mesma é adequada, diante da simplificação no tipo de especificação, pode ser difícil a aquisição do conjunto de regras que poderia controlar um sistema a contento, ou por não se ter acesso à fonte básica do conhecimento (não existe um especialista de fato), ou porque o conhecimento não é facilmente expresso em termos de regras de produção proposicionais.

Torna-se então fundamental uma investigação para diferentes tipos de representações de conhecimento, que possam ser adequadas a um leque maior de aplicações. Tal investigação deve sem dúvida levar em conta o processamento de incerteza, de modo mais generalizado que o possibilitado pelo uso da lógica nebulosa (inclusive), e que possa transformar a elicitação do conhecimento em uma tarefa menos árdua para o engenheiro do conhecimento, dispondo de maiores recursos de linguagem, tais como por exemplo o uso de símbolos predicativos referentes a uma linguagem de ordem superior, a consideração de quantificadores tais como os de relevância ou importância de fatos e regras, a utilização de regras do tipo preditivo, que permitirá a utilização do conhecimento na antecipação de eventos, etc. Tais recursos permitirão uma melhor tradução do conhecimento humano em termos de estruturas manipuláveis. Do mesmo modo, a adição de tais recursos demandará a investigação de como os mesmos poderão ser manipulados de modo a gerar resultados.

Uma segunda área que poderá ser abordada no futuro diz respeito ao aprendizado em sistemas de tempo real. Dentro deste âmbito, será necessária uma re-elaboração de algumas estruturas, de modo que se possa incorporar o aprendizado ao sistema de processamento de conhecimento. Este aprendizado poderá ser do tipo passivo, quando se deseja apenas efetivar uma aquisição do conhecimento onde a fonte do mesmo não é exatamente clara, e este aprendizado é feito *off-line*, como pode também ser do tipo ativo, quando o que se deseja não é realmente um aprendizado completo, mas uma sintonia fina em um determinado conhecimento, de modo que o mesmo se adapte a condições muito particulares de uma determinada aplicação, que não podem ser feitos *off-line*. Neste caso, o sistema deve se adaptar em tempo de execução, de modo a atingir o ponto onde encontra seu melhor desempenho.

Como pode-se concluir, "foi grande o caminho andado, mas ainda é longa a jornada".

Diante da conclusão deste trabalho e dos desafios de sua continuidade, espera-se que o mesmo possa ter contribuído, ou que possa algum dia contribuir, para o aperfeiçoamento da tecnologia existente, e de algum modo beneficiar a humanidade no que se refere a um aumento de eficiência dos processos, de modo que o conforto da tecnologia moderna possa servir para ampliar o bem estar de cada ser vivente.

# 9. REFERÊNCIAS BIBLIOGRÁFICAS


[1] Aho, A. V. ; Hopcroft, J. E. ; Ullman, J. D. - "Data Structures and Algorithms" - Addison-Wesley Publisshing Company, 1985.

[2] Aiello, L.; Cecchi, C.; Sartini, D. - "Representation and Use of Metaknowledge" - Proceedings of the IEEE, vol 74, nº 10, October 1986.

[3] Anderson, J. R. - "A Theory of the Origins of Human Knowledge" - Artificial Intelligence 40 (1989) 313-351.

[4] Aoki, H. ; Sasaki, K. - "Group Supervisory Control System Assisted by Artificial Intelligence" - Elevator World, February 90.

[5] Barney, G. C ; dos Santos, S. M. - "Elevator Traffic Analysis Design and Control" - IEE Control Engineering Series 2 - Peter Peregrinus Ltd, London UK. 2nd. Edition (1985).

[6] Booker, L. B. ; Goldberg, D.E. ; Holland, J. H. - "Classifier Systems and Genetic Algorithms" - Artificial Intelligence 40 (1989) 235-282.

[7] Chang, C.L. ; Lee, R. C. T. - "Symbolic Logic and Mechanical Theorem Proving".

[8] Chen, S. M.; Ke, J. S.; Chang, J. F. - "Knowledge Representation Using Fuzzy Petri Nets" - IEEE Trans. Knowledge and Data Eng., vol 2, nº 3, Sep. 1990.

[9] Enderton, H. - "A Mathematical Introduction to Logic".

[10] Fahlman, S. E. ; Hinton, G. E. - "Connectionist Architectures for Artificial Intelligence" - IEEE Computer, January 1987.

[11] Forgy, C. L. - "RETE: A Fast Algorithm for the Many Pattern/ Many Object Patttern Match Problem" - Artificial Intelligence 19 (1982) 17-37.

[12] Funabashi, M.; Mori, K. - "Knowledge Based Control Systems and Software for Building Expert Systems - 'EUREKA-II' " - Hitachi Review vol 37 (1988) nº 4.

[13] Gallant, S. I. - "Connectionist Expert Systems" - Communications of the ACM - February 1988, vol 31 nº 2.

[14] Goff, K. H. - "Artificial Intelligence in Process Control" - Mechanical Engineering, October 1985.

[15] King, M. S. ; Brooks, S. L. ; Schaefer, R. M. - "Knowledge-Based Systems" - Mechanical Engineering, October 1985.

[16] Laffey, T. J.; Cox, P. A.; Schmidt, J. L.;Kao, S. M.; Read, J. Y. - "Real-Time Knowledge-Based Systems" - AI Magazine, SPRING 1988.

[17] Lee, C. C. - "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I" - IEEE Trans. Syst. Man and Cybern., vol 20, nº 2, March/April 1990.

[18] Lee, C. C. - "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part II" - IEEE Trans. Syst. Man and Cybern., vol 20, nº 2, March/April 1990.

[19] Lesser, V. R.; Pavlin, J.; Durfee, E. - "Approximate Processing in Real-Time Problem Solving" - AI Magazine, SPRING 1988.

[20] Lim, M. H.; Takefuji, Y. - "Implementing Fuzzy Rule-Based Systems on Silicon Chips" - IEEE Expert, Feb. 1990.

[21] Looney, C. G.; Alfize, A. R. - "Logical Controls via Boolean Rule Matrix Transformations" - IEEE Trans. Syst. Man and Cybern., vol 17, nº 6, Nov/Dec 87.

[22] Looney, C. G. - "Fuzzy Petri Nets for Rule Based Decisionmaking" - IEEE Trans. Syst. Man and Cybern., vol 18, nº1, Jan/Feb 88.

[23] Maeda, M.; Murakami, S. - "A Design for a Fuzzy Logic Controller" - Information Sciences 45, 315-330 (1988).

[24] Maeda, H.; Murakami, S. - "A Fuzzy Decision-Making Method and Its Application to a Company Choice Problem" - Information Sciences 45, 331-346 (1988).

[25] Mamdani, E. H. - "Application of Fuzzy Logic to Aproximate Reasoning Using Linguistic Synthesis" - IEEE Trans. Computer, vol C-26 nº 12, Dec. 1977.

[26] Mizumoto, M. - "Fuzzy Controls Under Various Fuzzy Reasoning Methods" - Information Sciences 45, 129-151 (1988).

[27] Mizumoto, M.; Zimmermann, H. J. - "Comparison of Fuzzy Reasoning Methods" - Fuzzy Sets and Systems 8 (1982) 253-283.

[28] Nilson, N. J. - "Principles of Artificial Intellingence" - Tioga Press, 1980.

[29] Post, S. ; Sage, A. P. - "An Overview of Automated Reasoning" - IEEE Trans. on Syst. Man and Cybern. vol 20, nº1 Jan/Feb 1990.

[30] Rowan, D. A. - "On-Line Expert Systems in Process Industries" - AI Expert, August 1989.

[31] Rutherford, D. A.; Bloore, G. C. - "The Implementation of Fuzzy Algorithms for Control" - Proceedings of the IEEE, April 1976.

[32] Sorrels, M. E. - "A Time-Constrained Inference Strategy for Real Time Expert Systems" - In Proceedings of the IEEE 1985 National Aerospace and Electronics Conference, 1336-1341. Washington D. C.:IEEE Computer Society.

[33] Stroustrup, B. - "What is Object-Oriented Programming ?" - IEEE Software May 1988.

[34] Sugeno, M. - "An Introductory Survey of Fuzzy Control" - Information Sciences 36, 59-83 (1985).

[35] Thistle, J. G. ; Wonham, W. M. - "Control Problems in a Temporal Logic Framework" - Int. Journal Control 1986 vol 44, nº 4, pg. 943-976.

[36] Umeda, Y. ; Uetani, K. ; Ujihara, H. ; Tsuji S. - "Fuzzy Theory and Intelligent Options" - Elevator World, July 89.

[37] Warfield, J. N. - "Binary Matrices in System Modelling" - IEEE Trans. Syst. Man and Cybern., vol SMC-3 nº 5, Sep. 73.

[38] Woods, W. A. - "Important Issues in Knowledge Representation" - Proceedings of the IEEE, vol 74, nº 10, October 1986.

[39] Wright, M. L.; Green, M. W.; Fiegl, G.; Cross, P. - "An Expert System for Real-Time Control" - IEEE Software, March 1986.

[40] Yamakawa, T. - "High-Speed Fuzzy Controller hardware Systems: The Mega-FIPS Machine" - Information Sciences 45, 113-128 (1988).

[41] Zadeh, L. A. - "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes" - IEEE Trans. Syst. Man and Cybern., vol SMC-3, nº 1 , Jan. 1973.