Artigo - SDAF - Uma Ferramenta para o Desenvolvimento e Teste de Sistemas Utilizando Lógica Nebulosa

Asbstract

Este trabalho apresenta o SDAF - Sistema de Desenvolvimento de Aplicações Fuzzy, desenvolvido pela HI-Tecnologia, que constitui-se de uma plataforma de desenvolvimento para a geração e validação de bases de conhecimento para controle de processos e aplicações congêneres, utilizando lógica nebulosa (fuzzy logic) como modelo de representação e manipulação deste conhecimento. Inicialmente alguns conceitos relacionados com fuzzy logic são apresentados, seguido da apresentação do SDAF como ambiente de programação, e da especificação da linguagem LEARN, utilizada pelo ambiente. Em seguência é feita uma descrição de como se desenvolve uma aplicação, utilizando o módulo depurador do ambiente e por fim é dado panorama da evolução prevista para o projeto.

Introdução

Como fruto das pesquisas feitas em inteligência artifical, a utilização de sistemas baseados em conhecimento (sistemas especialistas, linguagens de programação em lógica, etc.) no desenvolvimento de aplicações que envolvem sistemas computadorizados ou microprocessados, tem tido resultados que cada vez mais justificam todo o esforço empregado, tanto a nível de pesquisas como desenvolvimento, transformado em ferramental teórico já bem consolidado em sistemas que podem ser efetivamente utilizados em aplicações práticas.

As aplicações que mais se adaptam a este novo paradigma são justamente aquelas que não têm solução (ou não têm uma boa solução) dentro dos métodos clássicos da teoria de sistemas. Estas, normalmente apesar de serem perfeitamente compreendidas por especialistas humanos, têm uma difícil modelagem fora da linguagem natural, o que inviabiliza a aplicação de modelos, tais como, por exemplo, os de sistemas lineares.

Muitos sistemas baseados em conhecimento foram desenvolvidos nos últimos anos em aplicações industriais [1], e muito se aprendeu a respeito de suas qualidades e suas deficiências. Uma de suas qualidades mais óbvias é a que os modelos levantados se assemelham muito ao modo de raciocínio humano, o que facilita sua compreensão pelos técnicos que o manuseiam, permitindo que o mesmo seja modificado de modo a incorporar novas carcterísticas, ou mesmo tenham seu desempenho melhorado, com uma menor dose de esforço. Algumas deficiências levantadas foram a dificuldade de se efetuar sua integração com sistemas de controle (interface) e o fato de não tratar idéias vagas, imprecisas, com o mesmo nível de abstração com que um ser humano raciocina e consegue chegar a soluções.

Um dos últimos ferramentais matemáticos que apareceram de modo a sanar tais deficiências foi a utilização da teoria de conjuntos nebulosos (fuzzy sets), que veio a dar origem a lógica nebulosa (fuzzy logic) [2,3]. Com o uso da lógica nebulosa, os sistemas baseados em conhecimento conseguiram ganhar um novo recurso, que permite a representação e manipulação de conceitos mais abstratos, utilizados no vocabulário humano (tais como alto, baixo, grande, pequeno, etc), que a despeito de serem vagos, imprecisos, permitem a manipulação de idéias e a tomada de decisões[4]. Do mesmo modo, a lógica nebulosa propicia uma integração mais amigável entre os sinais que podem ser colhidos por sensores, e o significado que a eles é atribuído, a nível de raciocínio (inferência). O mesmo é válido para o inverso, ou seja, a transformação de decisões razoavelmente abstratas em sinais de controle, ou comandos.

Lógica Nebulosa

Algumas perguntas que se fazem importantes responder, dentro do contexto da utilização da lógica nebulosa na implementação de sistemas baseados em conhecimento são:

- O que é FUZZY LOGIC ?
- Para que serve ?
- Como utilizá-la ?

Lembrando-se que a meta da utilização de sistemas baseados em conhecimento humano possa ser representado por algum tipo de estrutura, e que estas estruturas possam ser manipuladas automaticamente por sistemas microprocessados, gerando mais conhecimento na forma de diagnóstico ou de decisões, e estes por sua vez aplicados efetivamente no controle de qualquer tipo de processo ou dispositivo, pode-se dizer que a lógica nebulosa é ao mesmo tempo, um meio de se representar conhecimento, que incorpora de certo modo o caráter abstrato de alguns conceitos utilizados pelo ser humano na tomada de decisões, e ao mesmo tempo é uma representação que permite sua manipulação por procedimentos automáticos, que transformam uma descrição de um comportamento do sistema, colocado na forma de regras, no comportamento efetivo do sistema.

O primeiro elemento importante que se destaca na lógica nebulosa é a definição e o uso de "variáveis Linguísticas". Variáveis linguísticas são, a princípio, os elementos simbólicos utilizados para descrever o conhecimento. Uma variável linguística é definida como sendo uma variável, que representa um parâmetro que se deseja manipular e que recebe como valor um conceito linguístico, que pode ser abstrato ou vago. Exemplos de conceitos linguísticos são, por exemplo, alto, médio, baixo, quente, frio, forte, fraco, tenso, relaxado, apertado, folgado, etc. A cada conceito linguístico, está associado um significado, na forma de um conjunto nebuloso. Um conjunto nebuloso pode ser definido como uma função, (chamada de função de pertinência), que tem por domínio um universo de valores, (chamado de universo de discurso), podendo o mesmo ser contínuo ou discreto. Esta função associa a uma grandeza qualquer representada no universo de discurso, uma medida da compatibilidade entre cada possível valor da grandeza, com o valor linguístico que está sendo representado pelo conjunto nebuloso.

Com uso de variáveis linguísticas e seus valores representados por nebulosos, pode-se implementar o que se chama de uma proposição nebulosa. Uma proposição nebulosa é simplesmente a atribuição de um valor linguístico, associado a um conjunto nebuloso que define seu significado, a uma variável linguística. Por exemplo, tomando-se uma variável linguística "temperatura 1", defini-se que a mesma pode assumir os seguintes linguísticos: baixa, média e alta, cada qual associado a um conjunto nebuloso que lhe dá significado. Pode-se então formar 3 tipos de proposição nebulosa com esta variável linguística:

- temperatura 1 é baixa
- temperatura 1 é média
- temperatura 1 é alta

Com o uso de proposições nebulosas, pode-se então montar a estrutura básica de representação do conhecimento em lógica nebulosa, que são as regras nebulosas. Regras nebulosas são basicamente relações de implicação, onde representa-se que se uma determinada condição é satisfeita, então uma conclusão pode ser tomada. Em lógica nebulosa a condição não precisa ser associada a uma certeza, podendo ser descrita por uma proposição nebulosa, ou a conjunção de diversas proposições nebulosas, que por sua própria definição, contém elementos de incerteza. Uma regra nebulosa pode ser colocada, por exemplo, como a seguir:

SE temperatura 1 é alta E temperatura 2 é alta
ENTÃO refrigeração é aumentada

As proposições nebulosas que fazem parte da condição de regra são chamadas de antecedentes, e as que fazem parte da conclusão são chamadas de consequentes.

Os antecedentes de uma regra serão sempre proposições nebulosas do tipo perceptivo, ou seja constata-se uma determinada condição, condição esta que não necessita ser exprimida com clareza (pode ser vaga). Já os consequentes das regras podem ser de dois tipos. Podem também ser do tipo perceptivo, quando se atinge conclusões de um maior nível de abstração, como por exemplo, na seguinte regra:

SE temperatura 1 é alta E pressão 2 é muito alta
ENTÃO situação é perigosa

Outro tipo de consequente é o tipo decisivo, ou seja, algum tipo de decisão é tomada, diante da constatação de uma determinada condição, como por exemplo na seguinte regra:

SE situação é perigosa
ENTÃO abrir bastante a válvula de segurança

Com o uso dos elementos colocados na lógica nebulosa, a mesma pode ser utilizada na implementação de um sistema de controle. O conhecimento do especialista referente às relações entre grandeza abstrata é codificado na forma de regras, gerando uma base de regras, e em seguida cada conceito abstrato é codificado, tendo seu significado exprimido na forma de conjuntos nebulosos, que são codificados por meio de funções de pertinência definidas sobre os universos de discurso de cada grandeza manipulada. Estas funções são então armazenadas em uma base de funções. Diante da definição de uma base de regras e de uma base de funções que acumulam o conhecimento sobre um dado sistema, uma máquina de inferência

pode atuar, manipulando as medidas que são efetuadas diante de cada grandeza, interpretando estas medidas de acordo com o conhecimento expresso nas bases de regras e de funções, e gerando os sinais de controle que irão efetivamente controlar o sistema.

Esta breve introdução esclarece, em termos gerais, o que é FUZZY LOGIC , para que serve e como pode-se utilizar a mesma. Nas seções seguintes, serão desenvolvidos detalhes maiores sobre o ambiente SDAF, a linguagem utilizada pelo mesmo, e como devem ser desenvolvidas aplicações, diante dos recursos que o SDAF fornece, tais como o depurador simbólico e o editor de funções de pertinência.

O SDAF

O SDAF é constituído por dois módulos interconectados, operando em equipamentos distintos, e ligados ao processo a ser controlado (aplicação). Conforme mostrado na figura 01, estes módulos são:

- Ambiente Desenvolvido - PC
- Controlador FUZZY - Microcontrolador Industrial

Módulo do Ambiente de Desenvolvimento:

Este módulo foi projetado para rodar em microcomputador do tipo IBM/PC-XT/386, rodando sob sistemas operacionais compatíveis como o MS-DOS-V3.0. Ele possui uma interface homem-máquina amigável, dotada de recursos gráficos, menus, janelas de entrada de dados, help sensível a contexto, operando com ou sem mouse, etc.

O objetivo deste módulo é permitir o desenvolvimento de aplicações

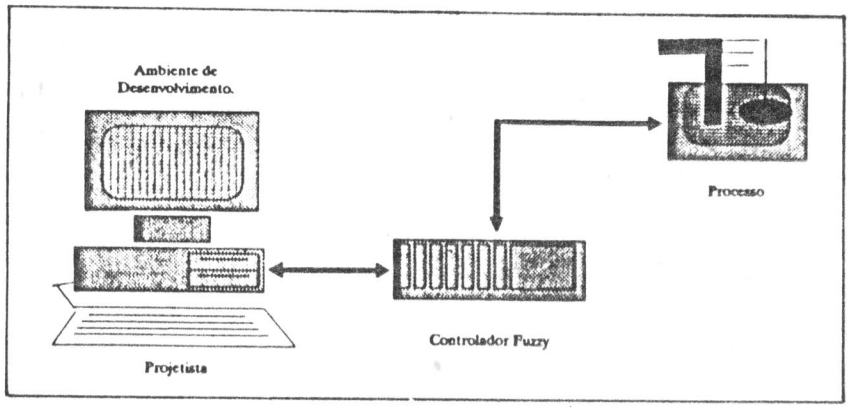

Figura 01 - Módulos do SDAF

que utilizem controladores baseados em regras fuzzy, e para tanto oferece, entre outros, os seguintes recursos:

- Linguagem de representação de conhecimento utilizando regras nebulosas;
- Editor de funções de pertinência;
- Gerador de código para o Controlador Fuzzy;
- Mecanismos para programação dos canais de comunicação com o processo;
- Operação remota do controlador fuzzy;
- Depurador Simbólico On line;

Controlador "FUZZY":

Este módulo foi implementado sob uma plataforma de hardware baseada no micro-controlador industrial MCI da HI Tecnologia. A justificativa para a implementação deste módulo em outra máquina, advem da necessidade de conexão física com o processo a ser controlado. Este equipamento possui todos os recursos necessários para integração com um processo real, tais como, entradas e saídas analógicas/digitais, interface com o operador e rede local de comunicação.

A figura 02 apresenta um diagrama lógico ilustrando a organização do software do controlador fuzzy, conforme implementado no referido equipado.

Opcionalmente, este módulo dispõe de uma interface de operação local, permitindo ao usuário comandar a execução do programa localmente. Existem comandos para ativar, suspender, executar passo a passo as regras, medir o tempo que o processador gasta para executar as instruções, etc..

Desenvolvimento de Projetos

O desenvolvimento de sistemas de controle baseado em lógica fuzzy no SDAF, é um processo interativo, composto da seguintes etapas:

- Identificação das variáveis de entrada/saída do processo;
- Definição das partições de cada variável linguística do sistema;
- Edição das funções de pertinências associadas a cada termo linguístico previamente definido;
- Edição de regras nebulosas;
- Compilação do programa;
- Transferência da base de conhecimento do ambiente de desenvolvimento para o controlador fuzzy;

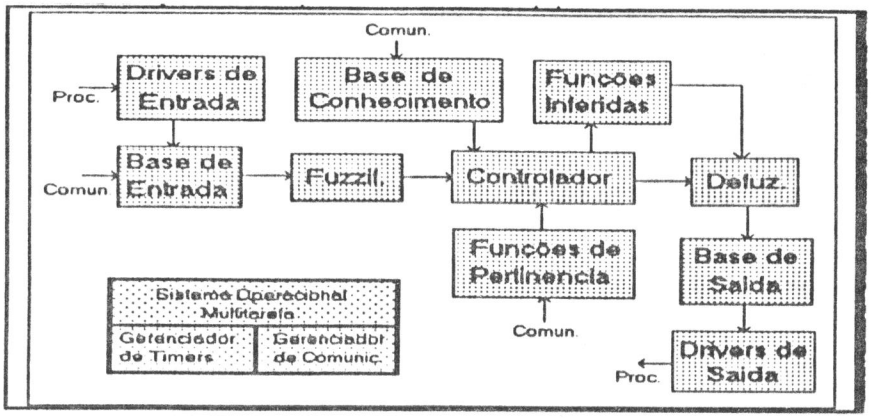

Figura 02 - Software do Controlador Fuzzy

- Execução/Depuração do programa;
- Análise de desempenho do sistema;
- Alteração nas regras do programa e/ou nas funções de pertinência, até o sistema atingir desempenho desejado.

Para facilitar a programação, o SDAF permite que as variáveis linguísticas de entrada/saída do sistema, sejam convenientemente escaladas no range equivalente ao do sensor, na unidade de engenharia correspondente.

O Editor de Funções

O SDAF dispõe de um editor gráfico que permite a edição/manipulação das funções de pertinência. As funções mais usuais, tais como, função sigmóide, triangular e gaussiana, são particularmente fáceis de serem geradas. Outras funções podem ser geradas através de técnicas de interpolação, seja via gráficos ou tabelas. A figura 03 apresenta uma tela típica do editor de funções, tal como implementado na versão atual do sistema.

Uma vez definido o programa fuzzy, o SDAF automaticamente transporta as partições associadas às variáveis linguísticas definidas no programa para o editor de funções, para que sejam configuradas pelo usuário.

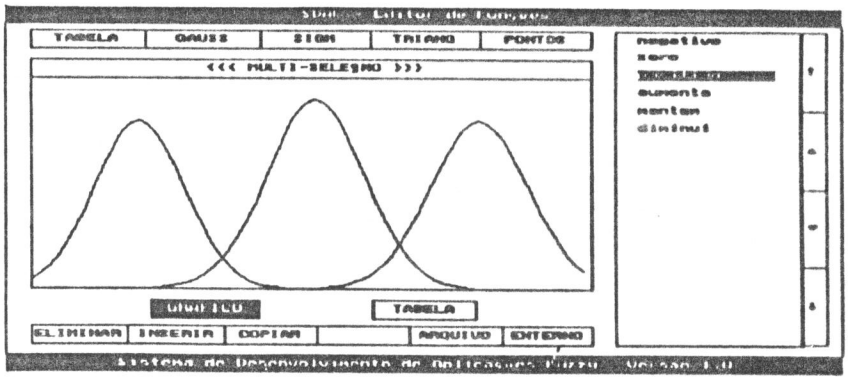

Figura 03 - Editor de Funções de Pertinências

A Linguagem LEARN

O ponto de partida para a utilização do SDAF, é a linguagem LEARN - Linguagem de Especificação de Aplicações mediante Regras Nebulosas, utilizada pelo sistema para a implementação das bases de regras nebulosas. Esta linguagem foi especialmente concebida para ser utilizada dentro do contexto deste sistema. Os recursos a ela adicionados se limitaram a um conjunto mínimo que permite o processamento de regras nebulosas, não encontrados nas linguagens de alto nível.

A Sintaxe da Linguagem LEARN

O formato geral dos programas escritos nesta linguagem é o seguinte:

- Declaração das variáveis linguísticas:
- Diretiva de Início de programa;
- Corpo do programa;
- Diretiva de fim de programa;

A declaração das variáveis linguísticas seguem a seguinte sintaxe:

VAR Nome – VAR [Universo – Discurso] = [Conjunto – Partições]

Onde:

VAR	Diretiva para a declaração de variáveis;
Nome_Var	Nome da variável linguística (ou variável *fuzzy*) a ser declarada;
Universo_Discurso	Define o domínio numérico da variável *fuzzy*;
Conjunto_Partições	Conjunto de funções que descrevem o "grau" de pertinência da assertiva em cada um dos pontos do respectivo universo de discurso;

Um Programa na Linguagem LEARN

A figura 04 apresenta um exemplo de um pequeno programa escrito na linguagem LEARN. Observe a sintaxe das regras, que é sempre do tipo:

SE Condição–1 E Condição–2 ...ENTÃO Ação;

As condições são expressões simples do tipo:

Var–1 = = Partição–i (Onde Partição–i deve ser uma partição da variável var–1).

```
#Programa simplificado para controle de nível de tanques
VAR  ANALOGICA Erro[0, 100] = { Negativo, Zero, Positivo};
     ANALOGICA Valvula[0, 200] = {Diminui, Mantem, Aumenta};
ENTRADA Erro;
SAIDA Valvula;
REGRAS
GRUPO Principal
{
# Inicio das Regras
   SE Erro = = Negativo ENTAO Valvula = Aumenta;
   SE Erro = = Zero ENTAO Valvula = Mantem;
   SE Erro = = Positivo ENTAO Valvula = Diminui;
}
FimGrupo
FimRegras
```

Figura 04 - Exemplo de um programa na linguagem LEARN

Tipos de Variáveis

As variáveis do sistema foram inicilamente concebidas para suportar aplicações de controle em tempo real, e por isso podem ser dos seguintes tipos:

- Variáveis analógicas: São variáveis numéricas de uso geral, podem ser de entrada, de saída, ou interna.
- Variáveis digitais: São variáveis lógicas de uso geral, também podem ser de entrada, de saída, ou interna.
- Timers analógicos: São variáveis numéricas internas do sistema, que possuem a característica adicional de serem atualizadas automaticamente pelo sistema, à medida que o tempo passa. Funcionam em sincronismo com o relógio do sistema.
- Timers digitais: São variáveis lógicas associadas a temporizadores do sistema, sua diferença em relação aos "timers analógicos" é que pode assumir apenas dois estados - "tempo expirado" e "tempo não expirado".

Desenvolvimento de Aplicações - O Depurador

A construção de uma base de conhecimento é normalmente um processo incremental, onde o conhecimento vai sendo levantado junto ao especialista em etapas. A cada etapa, um novo conjunto de regras é acrescentado à base, e o projetista do sistema, junto com o especialista, deve verificar a consistência destas regras, validando o comportamento do sistema.

Uma das técnicas na elaboração de uma base de conhecimento, que facilita a própria elicitação do conhecimento e o desempenho computacional do sistema em ambiente de tempo real é a utilização de bases de regras estruturadas, ou modularizadas. O SDAF permite a construção de bases deste tipo por meio da definição de grupos. Cada grupo deve conter, portanto, uma pequena parte do conhecimento, relativa a uma determinada especialização, em um mecanismo conhecido como foco de atenção. Um dos grupos, normalmente o grupo principal, deve conter o meta-conhecimento, ou seja, o conhecimento do conteúdo dos diversos grupos, e de quando acessá-los. O SDAF permite que este acesso seja feito de duas maneiras. A primeira delas, é a execução uma única vez de um determinado grupo, processada pelo comando EXECUTA. Este tipo de acesso é feito quando se deseja apenas uma aplicação de um procedimento, de modo semelhando a uma chamada de função em programação convencional. O segundo modo de se acessar um grupo é por meio do comando CHAVEIA, quando sob determinadas condições levantadas por algumas regras do grupo, determina-se que aquele grupo já não é mais o adequado a ser processado, e com isso se deseja efetivamente chavear o processamento para um outro grupo de regras, que estão mais adequados à situação corrente do sistema. Para o desenvolvimento de uma base de conhecimento, o procedimento mais usual é a determinação de um grupo principal, onde sejam colocadas as diretrizes gerais de atuação, e a cada etapa, sejam acrescentados novos grupos de regras, que podem então ser detahados pelo projetista, que passa a avalidar o funcionamento do grupo, efetuando as correções que forem pertinentes.

Durante este processo, é necessário que o projetista disponha de alguns recursos adicionais, as ferramentas de depuração, que o auxiliam na avalização do comportamento do sistema. Este conjunto de recursos é reunido no SDAF, no

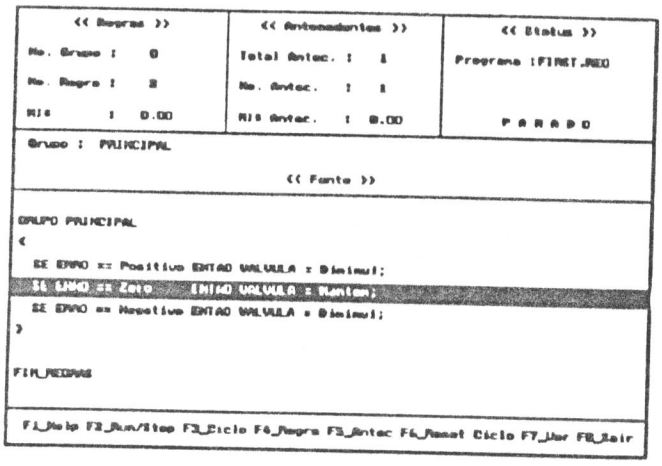

Fig.05-a Tela de fonte do Depurador

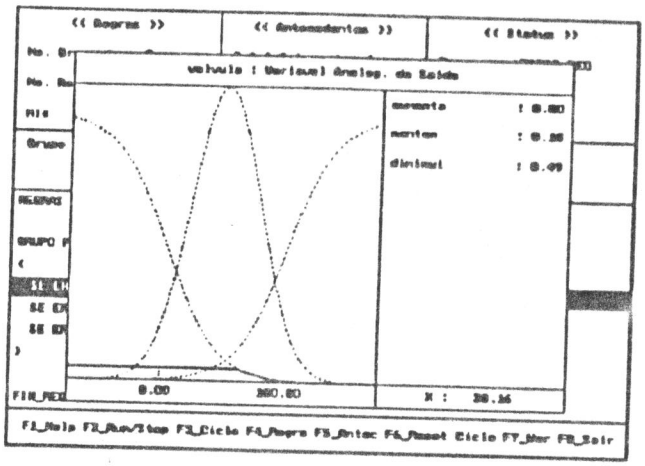

Fig.05-b Função inferida-Depurador

módulo depurador, vide figura 05.

O módulo depurador do SDAF permite o controle da execução das regras, permitindo o disparo e a parada da máquina de inferência em qualquer ponto do programa. Com isso, consegue-se executar as regras passo a passo sendo executado, como o estado do sistema após a execução do mesmo. Este passo pode ser a nível de ciclo, regra ou antecedente. A nível de ciclo, a máquina executa um ciclo de inferência e pára, esperando um próximo comando do operador. A nível de regra a máquina executa uma determinada regra ao comando do operador, parando em seguida, e a nível de antecedentes, a máquina para após executar cada antecedente, exibindo o estado do sistema. Um outro comando auxiliar é colocado de modo a reinicializar as variáveis do sistema.

Outro recurso importante do depurador é o processo pelo qual se pode analisar o conteúdo das diversas variáveis utilizadas. Com este, permite-se verificar o estado de cada variável. Caso sejam variáveis digitais, mostra-se seu estado lógico (FALSO/VERDADEIRO). Caso sejam variáveis analógicas de entrada, mostra-se o valor medido da variável, e para cada predicado linguístico associado à mesma, como o mesmo se relaciona diante do valor medido, dando uma idéia da compatibilidade entre o valor medido e o conceito associado ao predicado linguístico. Para variáveis analógicas de saída, mostra-se além do conjunto completo de conjuntos nebulosos associados a ela, o estado do conjunto nebuloso inferido , que dará origem ao valor defuzzyficado, que será enviado ao controle.

Utilizando-se o depurador, o projetista consegue efetuar o desenvolvimento incremental do sistema, pois, após a elicitação do conhecimento sobre a forma de regras e funções de pertinência, o mesmo pode ser validado executando-se o programa de regras passo a passo, o que permite a detecção de regras com comportamento inadequado, facilitando sua correção.

Conclusões e Perspectivas

O SDAF foi concebido como um sistema de suporte ao desenvolvimento de aplicações que utilizem a lógica nebulosa (fuzzy logic) como modelo para as ações de controle cabíveis. Para tanto, foi desenvolvida uma linguagem com o objetivo de construir modelos que estejam o mais próximo possível de uma linguagem natural, sobre a qual um especialista descreva seu raciocínio diante de um determinado estado do sistema. Esta linguagem incorpora as inovações possibilitadas pelo uso da lógica nebulosa como instrumento, que é uso de termos vagos e imprecisos representando conceitos. A representação e manipulação de termos deste tipo é o que diferencia basicamente o SDAF de outros shells para sistemas especialistas.

Neste trabalho, procurou-se apresentar a constituição básica do SDAF como uma ferramenta que permite ao projetista de sistemas implementar efetivamente sistemas baseados em conhecimento via lógica nebulosa.

O sistema, da maneira com que foi descrito, tem-se se mostrado adequado para um grande número de aplicações. Entretanto, para algumas destas observou-se que poderiam ainda ser acrescentados mais recursos à linguagem, sem comprometer a sua uniformidade, e permitindo a descrição de certas classes de conhecimento de forma mais simples. Em versões posteriores do SDAF, estas modificações, atualmente em testes, serão incorporadas à linguagem.

Dentre estas modificações, po-

citar o uso de operadores matemáticos nos antecedentes das regras, e o uso de variáveis matriciais. No ambiente, planeja-se um módulo também implementado um módulo de análise estatística destes sinais, auxiliando o trabalho de depuração e avaliação de desempenho do sistema. Outro recurso que está sendo implementado é um módulo de diálogo com o usuário, de modo que não só sinais de processo sejam utilizados, mas também o usuário possa intervir na execução do programa. Isso permitirá a utilização do SDAF na área de shells de apoio a tomada de decisões, abrangendo as áreas de diagnóstico médico, mecânico, análise de falhas, análise de mercado, investimento. etc. Em termos de hardware, planeja-se a construção de uma placa para PC, que incorpore o interpretador e a entrada/saída de sinais, em substituição ao Controlador Fuzzy. Além de funcionar de modo semelhante, esta placa possibilitará aos usuários não-industriais (que utilizarão o SDAF como um shell para tomada de decisões) uma configuração mais compacta e de menor custo, pois não serão necessárias certas características como o robustez, a imunidade a ruídos, elevada capacidade de i/o, etc., que são particularidades do meio industrial.

BIBLIOGRAFIA

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

[2] Zadeh, L.A. - "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes" - IEEE Trans. on Systems, Man and Cybernetics, vol SMC-3, nº 1, Jan 1973

[3]Sugeno, M. - "An Introductory Survey of Fuzzy Control" - Information Siences 36 59-83(1985).

[4] H. J. Zimmermann - "Fuzzy Sets, Decision Marking and Systems" Kluwer Academic Publishers, 1987.

Autores: Ricardo Ribeiro Gudwin
Maurício Alves da Silva
Hélio Jacques de Almeida Jr.
Isaias Mendes Campos Ribeiro
• Fernando Antonio Campos Gomide
•• Armando Rocha
• Prof.: Unicamp
•• FEE - Depto. Computação e Automação