UMA TÉCNICA COM LÓGICA NEBULOSA PARA VERIFICAÇÃO DE UM PROTOCOLO DE GERÊNCIA HIERÁRQUICA DE REDE

Orlando Bernardo Filho[1,2], Marcial Porto Fernandez[2]
Aloysio de Castro Pinto Pedroza[2,3], Jorge Lopes de Souza Leão[2]

(1) Departamento de Enga. de Sistemas e Computação/FEN-UERJ
E-mail: orlando@eng.uerj.br
(2) Programa de Engenharia Elétrica/COPPE-UFRJ
Caixa Postal: 68504 CEP: 21945-700 Rio de Janeiro - RJ
TEL: (021) 260-5010 FAX: (021) 290-6626
E-mail: orlando@gta.ufrj.br/marcial@gta.ufrj.br/aloysio@gta.ufrj.br/ leao@gta.ufrj.br
(3) Departamento de Eletrônica/EE-UFRJ

Resumo

Este trabalho apresenta uma técnica para verificação formal aplicada a um modelo proposto de gerência hierárquica de grandes redes de comunicação ATM. Tal técnica de verificação usa um Sistema de Lógica Nebulosa para estimar a função de busca heurística no grafo de execução do protocolo considerado, objetivando encontrar um estado global meta que satisfaça uma dada propriedade de interesse.

Abstract

This paper shows a protocol formal verification technique applied to verify a model proposed of a large management hierarchical systems in a large ATM communication network. This technique uses a Fuzzy Logic System to estimate a function heuristic search in a protocol execution graph in order to seek an objective global state to prove an interest property.

1 INTRODUÇÃO

Diversos trabalhos sobre verificação de protocolos de comunicação, baseados em modelos de estados, já foram propostos, abordando as mais variadas técnicas. A maior parte deles consiste em algum tipo de análise, envolvendo uma busca no espaço de estados do modelo considerado. A exploração completa desse espaço de estados seria suficiente para resolver de forma categórica o problema da verificação. No entanto, é sabido que diversos protocolos possuem uma árvore de alcançabilidade muito grande, provocando o esgotamento de recursos computacionais (tempo e/ou memória), o que inviabiliza a sua aplicação pura e simples.

O esgotamento dos recursos computacionais conduziu à proposição de algumas técnicas de verificação [1,2,3] que consistem, geralmente, em variações da idéia básica do método original (exploração do espaço de estados), objetivando uma redução do grafo de estados, durante a análise de alcançabilidade. Tais técnicas apresentaram resultados interessantes,

principalmente em especificações de protocolos de tamanho médio. Entretanto, para protocolos de redes muito complexas, a dificuldade permanece, sendo assim, alguns pesquisadores resolveram propor um outro grupo de métodos baseados em simulações [4,5,6].

Os métodos de verificação, baseados em simulações, não enfrentam o problema do esgotamento de recursos computacionais, pois esses têm como meta colocar o protocolo em execução sob condições normais. No trabalho de WEST [6], por exemplo, é mencionada como solução para a verificação de protocolos reais, uma pesquisa (simulação) aleatória. Contudo, sem fazer o armazenamento de pelo menos algumas possibilidades de execução, não há como garantir a existência de uma determinada propriedade no protocolo, pois esse pode percorrer sempre um mesmo caminho.

Neste artigo expõe-se uma solução alternativa para o problema da verificação de protocolos, valendo-se de uma busca heurística no seu grafo de execução com o emprego de Lógica Nebulosa. A busca do protocolo sob análise será guiada pelo conhecimento genérico da categoria de que esse protocolo pertence. Essa análise se assemelha à simulação, todavia a execução do protocolo é dotada de informação, o que evita tanto as buscas repetidas por certos caminhos quanto a explosão do espaço de estados.

Um protocolo de gerência de rede, usando um modelo hierárquico, foi escolhido como aplicação da técnica de verificação, visto que o problema da gerência de rede começa a ficar difícil de tratar a partir do momento em que o número de agentes do sistema aumenta, vindo a comprometer o processamento no gerente da rede. O modelo hierárquico proposto [7,8] apresenta vários gerentes que passam a ser, cada um deles, os responsáveis por um grupo de agentes. Em caso de falha de algum gerente, o seu superior hierárquico assume a gerência dos seus agentes.

Na seção seguinte, indroduzem-se os principais conceitos de Lógica Nebulosa. Na terceira seção mostra-se a técnica de busca heurística em espaço de estados a ser usada pelo método de verificação. Apresenta-se, na quarta seção, o método de verificação propriamente dito; enquanto, na quinta seção, é exposto o problema da gerência de uma rede de grandes dimensões. Por fim, na sexta seção, temos a técnica de verificação aplicada ao protocolo hierárquico de gerência de rede.

2 A LÓGICA NEBULOSA

A Lógica Nebulosa é uma aplicação da teoria dos conjuntos nebulosos. Na matemática clássica, define-se um conjunto como uma coleção de elementos distintos ou objetos que podem ser finitos ou não. Tal conjunto pode ser descrito de várias maneiras, entre as quais, destacam-se: a enumeração de cada um de seus elementos ($A = \{1,2,3,4\}$), ou uma condição de pertinência ($A = \{x \mid x \geq 5\}$).

Ao usar a descrição, a partir de uma condição de pertinência, quando um elemento x causar a avaliação dessa condição como verdadeira, esse pertence ao conjunto; caso a avaliação seja falsa, conseqüentemente, não fará parte do conjunto. Para caracterizar o valor-verdade da condição de pertinência, pode-se empregar uma função que retorna **1**, se essa condição for verdadeira, e **0** se for falsa.

Na teoria dos conjuntos nebulosos [9,10,11], criada por ZADEH [12] em 1965, a função de pertinência não retornará apenas aos valores **0** ou **1**, mas qualquer outro valor desse intervalo

[0,1], o que revela a existência de vários graus de pertinência. Sendo assim, um conjunto nebuloso $\tilde{A}$ possui o seguinte aspecto:

$$\tilde{A}=\{(x,\mu_{\tilde{A}}(x))|x\in X\} \quad (1)$$

O termo $\mu_{\tilde{A}}$ é a função de pertinência que mapeia o universo de discurso X ao espaço de pertinência M. Quando M possui apenas os dois pontos **0** e **1**, $\tilde{A}$ não é um conjunto nebuloso, sendo normalmente chamado de conjunto clássico.

Ao observar a definição de conjuntos nebulosos, percebe-se, que na verdade, esses podem ser vistos como um conjunto clássico de pares. O exemplo abaixo mostra o conjunto nebuloso $\tilde{A}$ dos números inteiros próximos de **4**.

$$\tilde{A} = \{(1,0.3), (2,0.6), (3,0.8), (4,1.0), (5,0.8), (9,0.6), ...\} \quad (2)$$

Como pode ser observado, **4** é o número mais próximo de **4**, logo ele recebe o índice **1.0** no conjunto nebuloso, enquanto os outros números que se afastam de **4** vão recebendo índices de menor valor. A função de pertinência que aparece explicitamente nos pares do conjunto do exemplo acima, pode ser fornecida analiticamente como é apresentado a seguir.

$$\tilde{A}=\{(x,\mu_{\tilde{A}}(x))|\mu_{\tilde{A}}(x)= (1+(x-4)^2)^{-1}\} \quad (3)$$

Dentro do estudo da Lógica Nebulosa e do raciocínio aproximado é bastante utilizada uma ferramenta conhecida como ***variável lingüística***, também chamada de variável de ordem mais alta. Essas variáveis não possuem números como valores, mas termos ou sentenças de uma linguagem natural ou artificial.

A idéia de se utilizar variáveis lingüísticas com valores de palavras ou frases de linguagens naturais ou artificiais está relacionada ao fato desses valores não serem tão específicos como números. Nesse sentido, nas ocasiões em que o tratamento de um valor numérico preciso é irrelevante para o resultado final da operação, como em controle de sistemas não lineares, por exemplo, a manipulação dos valores lingüísticos é mais fácil, pois estes estão em menor quantidade e são mnemônicos.

Uma variável lingüística é definida por uma quíntupla $(x,T(x),U,G,\tilde{M})$ onde:

- x é o nome da variável;
- $T(x)$ denota o conjunto de termos de x, isto é, o conjunto de nomes dos ***valores lingüísticos*** de x com cada valor sendo um conjunto nebuloso;
- U é o universo de discurso dos conjuntos nebulosos que formam os termos de $T(x)$;
- G é a regra sintática, que usualmente tem a forma de uma gramática, para gerar os nomes dos valores lingüísticos;
- $\tilde{M}(X)$ é a regra semântica que atribui um significado ao termo X do conjunto $T(x)$, ou seja, $\tilde{M}(X)$ é um subconjunto nebuloso de U.

Nos Sistemas de Lógica Nebulosa (SLN), representados pela figura 1, em geral, são fornecidas entradas precisas para um módulo ***fuzificador*** que, por sua vez, fornece parâmetros nebulosos para um motor de inferência, o qual processa a aplicação de uma regra do tipo *SE-ENTÃO*, constituída de proposições, envolvendo termos de variáveis lingüísticas. Após o

processamento de uma regra, o valor nebuloso, obtido como resposta da inferência, é ***desfuzificado***, obtendo-se, dessa forma, a saída precisa do sistema.

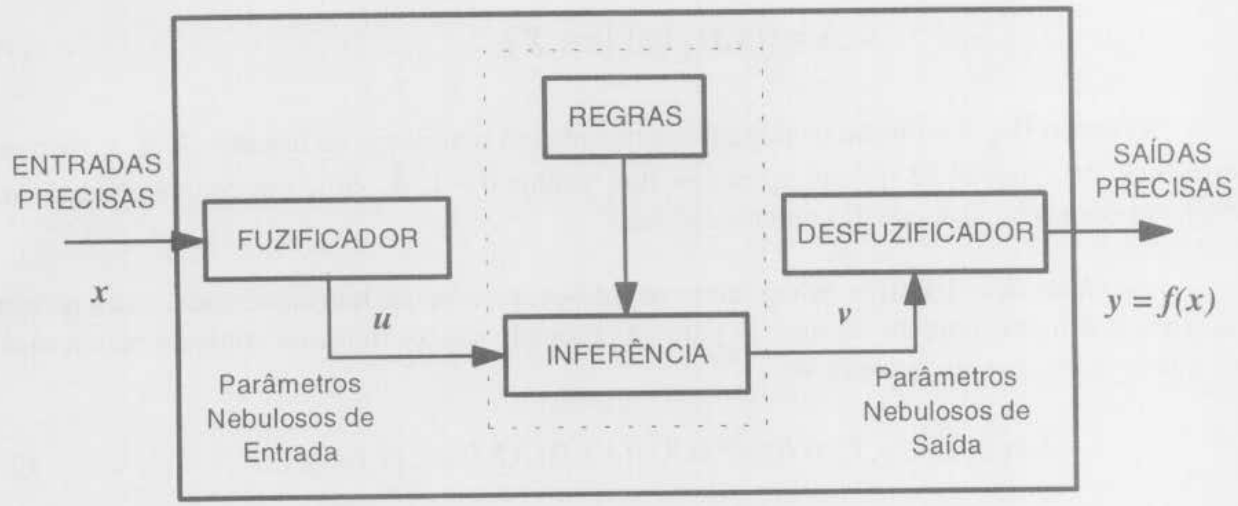

Figura 1: Sistema de Lógica Nebulosa.

Um parâmetro preciso pode ser ***fuzificado*** seguindo um método simples, trapezoidal, triangular ou gaussiano; já a ***desfuzificação*** pode ser feita através do valor máximo, média de valores máximos ou por meio do cálculo do centróide. No que diz respeito à inferência, essa pode seguir a regra de ***modus ponens generalizado*** ou composição de relações nebulosas.

Relações nebulosas são subconjuntos nebulosos do produto cartesiano $X \times Y$. Essas relações possuem um papel importante dentro da teoria de conjuntos nebulosos, sendo aplicadas na resolução de diversos problemas. A definição das relações nebulosas é dada a seguir:

Sejam $X, Y \subseteq R$ os conjuntos dos universos de discursos, então

$$\tilde{R} = \{((x,y), \mu_{\tilde{R}}(x,y)) | (x,y) \in X \times Y\} \quad (4)$$

é chamada de uma relação nebulosa em $X \times Y$.

Uma das operações mais úteis definidas sobre as relações nebulosas é a chamada composição **max-min** que combina relações nebulosas de produtos de espaços diferentes.

Sejam $\tilde{R}_1(x,y), (x,y) \in X \times Y$ e $\tilde{R}_2(y,z), (y,z) \in Y \times Z$ duas relações nebulosas. A composição max-min entre $\tilde{R}_1$ e $\tilde{R}_2$ é dada pelo seguinte conjunto nebuloso:

$$\tilde{R}_1 \circ \tilde{R}_2 = \{[(x,z), \max_y \{\min\{\mu_{\tilde{R}_1}(x,y), \mu_{\tilde{R}_2}(y,z)\}\}] | x \in X, y \in Y, z \in Z\} \quad (5)$$

O termo $\max_y \{\min\{\mu_{\tilde{R}_1}(x,y), \mu_{\tilde{R}_2}(y,z)\}\}$ é a função de pertinência da composição de relações nebulosas.

Uma idéia básica na Lógica Nebulosa é a de que uma proposição **P** em uma linguagem natural ou artificial pode ser vista como uma coleção de restrições elásticas [13], restringindo os valores de uma coleção de variáveis $x = (x_1, x_2, ..., x_n)$ definidas nos universos U^n. Geralmente as variáveis, bem como, as restrições aparecem em **P** de maneira implícita. Visto nessa

perspectiva, a representação do conhecimento de **P** consiste, na essência, no processo de tornar explícitas as variáveis e restrições. Isso é obtido, representando **P** através de sua forma canônica $\mathbf{P} \rightarrow x$ É A, onde A é um predicado nebuloso.

3 TÉCNICA DE BUSCA HEURÍSTICA

As técnicas de busca Heurística podem ser usadas na verificação de protocolos de comunicação, a fim de otimizar a expansão do seu grafo de execução através da aplicação de critérios, reduzindo os caminhos para a obtenção de objetivos que provem uma dada propriedade de interesse do protocolo.

Neste trabalho, o algoritmo **A*** foi escolhido como base para a técnica de verificação, pois ele pode fornecer o melhor caminho para o nó objetivo, caso a função heurística utilizada busque o ponto ótimo. Isso é importante, na medida em que os nós que compõem os grafos dos protocolos de comunicação, normalmente, possuem uma grande quantidade de informação, necessitando, portanto, de muito espaço em memória/disco para armazenamento, podendo suscitar um maior tempo no processamento da verificação. A seguir, é apresentado em pseudo-código, o algoritmo **A*** [14].

```
1    ALGORITMO A*

2    { seja f' = g + h' a estimativa da função custo f que um nó x possui          }
3    { para atingir o nó meta                                                      }
4    { g é o custo do nó raiz ao nó x e h' é o custo que se espera de x ao nó meta }

5    VARIÁVEIS:
6    ESTRUTURA NÓ_SUC =      pSuc        : PONTEIRO PARA NÓ;
7                            próximo     : PONTEIRO PARA NÓ_SUC;
8    FIM {ESTRUTURA NÓ_SUC}

9    ESTRUTURA NÓ =    dados       : tipo_DADOS_do_nó;
10                     g           : INTEIRO;
11                     h'          : INTEIRO;
12                     ptrSuc      : PONTEIRO PARA NÓ_SUC;
13                     ptrPai      : PONTEIRO PARA NÓ;
14                     próximo     : PONTEIRO PARA NÓ;
15   FIM {ESTRUTURA NÓ}

16   AcheiObjetivo                  : BOOLEANO;
17   VELHO, MELHOR_NÓ, NÓ_RAIZ      : PONTEIRO PARA NÓ;
18   primABERTO, primFECHADO        : PONTEIRO PARA NÓ;
19   SUCESSORES                     : PONTEIRO PARA NÓ_SUC;

20   INÍCIO
21   Ler (NÓ_RAIZ);
22   AcheiObjetivo <- FALSO;
23   primFECHADO <- nil;
24   primABERTO <- NÓ_RAIZ;
25   NÓ_RAIZ^.g <- 0;
26   NÓ_RAIZ^.próximo <- nil;
27   NÓ_RAIZ^.ptrSuc <- nil;
28   REPETIR
29   MELHOR_NÓ <- primABERTO;
```

```
30    primABERTO <- primABERTO^.próximo;
31    MELHOR_NÓ^.próximo <- primFECHADO;
32    primFECHADO <- MELHOR_NÓ;
33    AcheiObjetivo <- É_Objetivo (MELHOR_NÓ);
34    SE NÃO AcheiObjetivo
35    ENTÃO
36    SUCESSORES <- GerarSuc (MELHOR_NÓ);
37    ENQUANTO SUCESSORES # nil FAÇA
38    TRATA_SUCESSORES (SUCESSORES^.pSuc);
39    SUCESSORES <- SUCESSORES^.próximo;
40    FIMENQUANTO
41    FIMSE
42    ATÉ AcheiObjetivo OU (primABERTO = nil);
43    FIM {A*}

44    SUBALGORITMO TRATA_SUCESSORES (Suc : PONTEIRO PARA NÓ);

45    VARIÁVEIS:
46    Estou_em_aberto, Estou_em_fechado : BOOLEANO;

47    INÍCIO
48    Estou_em_aberto <- FALSO;
49    Estou_em_fechado <- FALSO;
50    Suc^.ptrPai <- MELHOR_NÓ;
51    Suc^.g <- MELHOR_NÓ^.g + 1;
52    VELHO <- PEGAR_NÓ_IGUAL_NA_FILA (ABERTO,Suc);
53    SE VELHO # nil
54    ENTÃO
55    Estou_em_aberto <- VERDADEIRO;
56    Inserir_nos_sucessores (MELHOR_NÓ,Suc);
57    SE Suc^.g < VELHO^.g
58    ENTÃO
59    VELHO^.g <- Suc^.g;
60    VELHO^.ptrPai <- MELHOR_NÓ;
61    FIMSE
62    FIMSE
63    SE NÃO Estou_em_aberto
64    ENTÃO
65    VELHO <- PEGAR_NÓ_IGUAL_NA_FILA (FECHADO,Suc);
66    SE VELHO # nil
67    ENTÃO
68    Estou_em_fechado <- VERDADEIRO;
69    Inserir_nos_sucessores (MELHOR_NÓ,Suc);
70    SE Suc^.g < VELHO^.g
71    ENTÃO
72    VELHO^.g <- Suc^.g;
73    VELHO^.ptrPai <- MELHOR_NÓ;
74    Atualizar_custo_dos_sucessores (VELHO);
75    FIMSE
76    FIMSE
77    FIMSE
78    SE NÃO Estou_em_aberto E NÃO Estou_em_fechado
79    ENTÃO
80    Inserir_nos_sucessores (MELHOR_NÓ,Suc);
81    Colocar_ordenado_em_aberto (Suc);
82    FIMSE
FIM {TRATA_SUCESSORES}
```

4 TÉCNICA PARA A VERIFICAÇÃO DE MODELO DE ESTADO

A técnica para verificação de Modelos de Estados aqui apresentada, consiste em uma busca heurística no grafo de execução do protocolo sob estudo, objetivando encontrar um nó que acarretaria a correção de uma dada propriedade de interesse. Tal busca é feita segundo o algoritmo **A***, modificado para operar com variáveis lingüísticas da Lógica Nebulosa.

No algoritmo **A***, mostrado na seção anterior, o passo (ou subalgoritmo) da linha 81 **Colocar_ordenado_em_aberto (Suc)** tem por objetivo armazenar os nós recentemente criados na fila de nós abertos. A fila de *ABERTO*, no algoritmo **A*** original, mantém os nós ordenados de maneira crescente, segundo os valores de **f' = g + h'**, de tal forma que o primeiro nó da fila é aquele que possui o menor custo para o objetivo. Na técnica ora proposta, a fila de *ABERTO* será ordenada por intermédio do processamento de regras de um Sistema de Lógica Nebulosa (SLN).

O bom desempenho do algoritmo **A*** está calcado nas estimativas para o termo **h'** da função custo **f'**. A estimativa de **h'** é feita a partir do conhecimento específico do problema em análise. No caso em questão, deve-se adotar o conhecimento genérico sobre protocolos de comunicação.

A maior dificuldade presente na utilização do algoritmo **A***, é estabelecer a estimativa do valor de **h'** em cada nó, devido à eventual complexidade de tradução do conhecimento específico relacionado ao problema tratado; e, à sistematização (padronização) imprecisa desse conhecimento para determinados sistemas.

O emprego da Lógica Nebulosa facilita a descrição do conhecimento dos protocolos de comunicação a ser utilizado pelo algoritmo **A***, pois trata-se de um arcabouço matemático próprio para lidar com raciocínio aproximado.

A técnica de verificação será aplicada em um determinado protocolo de comunicação, doravante Sistema sob Estudo (**SE**), o qual deverá ser especificado por uma Técnica de Descrição Formal que se baseie em um modelo de estados e transições.

A especificação de **SE** deverá ser comparada à especificação de um outro sistema, aqui denominado Sistema Padrão (**SP**), considerado como um "bom" paradigma para **SE**. Pode-se entender **SP** como uma base de conhecimento genérica para auxiliar o projeto dos protocolos de comunicação.

A técnica prevê a utilização de algumas versões de **SP**, ou seja, vários paradigmas de diversas categorias de protocolos, a fim de que o usuário de um verificador, calcado nessa mesma técnica, possa definir qual dos referidos paradigmas mais se assemelha ao seu **SE**. Portanto, é a partir do estabelecimento das semelhanças entre os objetos do **SE** e do **SP**, feito pelo usuário de forma nebulosa, que começa o emprego da técnica de verificação. Em outras palavras, tem-se:

- $\mathbf{E} = \{e_1, e_2, ..., e_{n1}\}$ é o conjunto (clássico) dos estados parciais de **SE**;
- $\mathcal{E} = \{\epsilon_1, \epsilon_2, ..., \epsilon_{n2}\}$ é o conjunto (clássico) dos estados parciais de **SP;**

A partir de então, define-se a relação nebulosa: $\tilde{R}_{E\mathcal{E}}(e,\epsilon)$ que caracteriza a medida de semelhança entre os estados parciais do **SE** e do **SP**. Além dessa relação e da especificação do **SE**, é dado também um nó objetivo η_o a ser alcançado pelo algoritmo **A*** nebuloso ($\tilde{\mathbf{A}}$*) a partir do nó raiz η_R do grafo de execução de **SE**.

Uma vez que **SE** não possui, a princípio, a mesma quantidade de estados que **SP**, isto é, os sub-índices ***n1*** e ***n2*** são diferentes entre si, então para se obter um estado parcial e em **SP** que corresponda a um estado parcial **e** em **SE**, deve ser feita a seguinte composição de relações nebulosas:

$$\tilde{O}(e) = \tilde{O}(\mathbf{e}) \circ \tilde{R}_{E\mathcal{E}}(e,e) \qquad (6)$$

onde $\tilde{\mathbf{O}}(\mathbf{e})$ é um dos estados do **SE**, $\mathbf{e_i}$, ***fuzificado*** pelo método simples; e, $\tilde{O}(e)$ é o estado parcial nebuloso em **SP,** equivalente a $\tilde{\mathbf{O}}(\mathbf{e})$ em **SE**, que, após ser ***desfuzificado*** pelo método do máximo, fornece o objeto e_4 pertencente ao conjunto $\mathcal{E}$.

A equação (6) deve ser aplicada para todos os estados parciais do **SE** a fim de se obter os seus equivalentes em **SP**. Todavia essa equivalência, em geral, não será perfeita e, portanto, deverá ser medida para compor as regras de inferência de um SLN. A medida da equivalência dar-se-á pela variável lingüística ***semelhança***, cujo universo de discurso **U** é composto pelos valores da função de pertinência da relação $\tilde{R}_{E\mathcal{E}}(e,e)$. Na figura 2, pode-se observar a definição dessa variável lingüística.

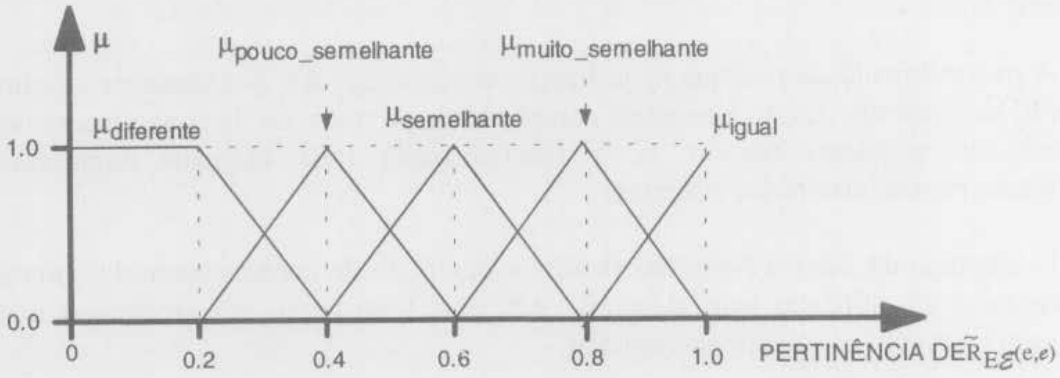

Figura 2: Funções de pertinência dos termos da variável lingüística ***semelhança***.

A medida da equivalência é fundamentalmente importante, pois a mesma pode ficar ainda mais prejudicada devido ao agrupamento dos estados parciais para compor os estados globais. Suponha que o **SE** e o **SP** tenham duas entidades comunicantes e que um dado estado global de **SE** seja o par $(\mathbf{e_1},\mathbf{e_2})$. Após aplicar-se a fórmula (6) para cada um dos estados parciais $\mathbf{e_1}$ e $\mathbf{e_2}$, chegou-se a conclusão de que os seus equivalentes, deveriam ser e_3 e e_5 no **SP**. No entanto, o par (e_3, e_5) não constitui um estado global em **SP** o que impediria a obtenção de um custo para o nó objetivo em **SP**. Logo, deve-se descartar os valores previamente ***desfuzificados*** e_3 ou e_5, e pegar como equivalentes do **SE**, os próximos estados parciais de **SP** com valores de pertinência imediatamente abaixo do máximo e assim por diante até se obter um par de estados parciais em **SP** que seja um estado global e ainda seja o mais próximo de $(\mathbf{e_1},\mathbf{e_2})$.

O subalgoritmo da linha 81 **Colocar_ordenado_em_aberto (Suc)** do algoritmo $\tilde{\mathbf{A}}$* aloca os nós na fila de *ABERTO* a partir de um SLN que manipula termos de uma outra variável lingüística, chamada ***posição*** [13], cuja definição aparece esquematicamente na figura 3. Nessa figura, observa-se o universo de discurso da variável caracterizado pelo percentual de nós na fila de *ABERTO*, pois, conforme $\tilde{\mathbf{A}}$* evolui, a quantidade de estados globais obtidos aumenta, sendo então necessário, inicialmente, ***desfuzificar*** um percentual para depois calcular a posição precisa atual.

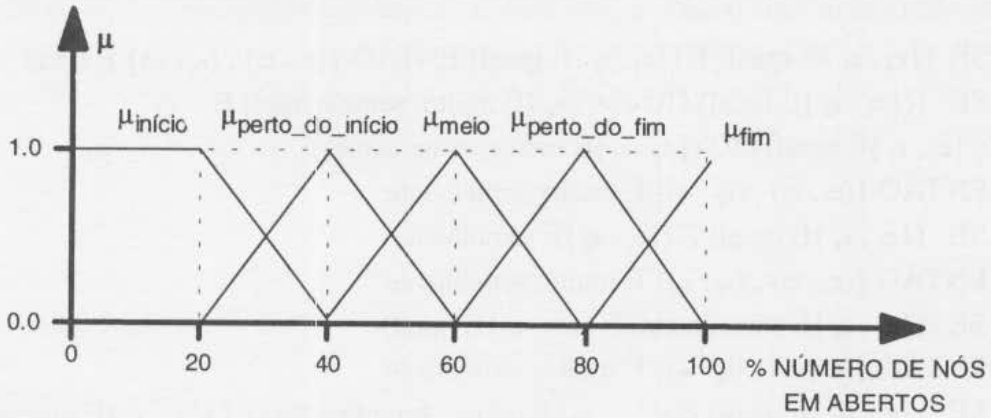

Figura 3: Funções de pertinência dos termos da variável lingüística ***posição***.

Além das variáveis lingüísticas ***semelhança*** e ***posição***, é definida também a variável ***custo*** (ver figura 4), que atribui termos nebulosos ao custo do nó do **SP,** equivalente ao nó **Suc** (um dos sucessores do nó em análise) do **SE**. O universo de discurso da variável ***custo*** é um conjunto cujos elementos são constituídos do número de passos de computação (ou disparo de transições). Uma vez que o número de passos de computação máximo de **SE** (sem repetições) não é conhecido, foi escolhido, como valor final do intervalo que compõe o universo de discurso de **U**, o maior caminho (sem repetições) de **SP** mais 60%, pois geralmente **SE** é maior que **SP**.

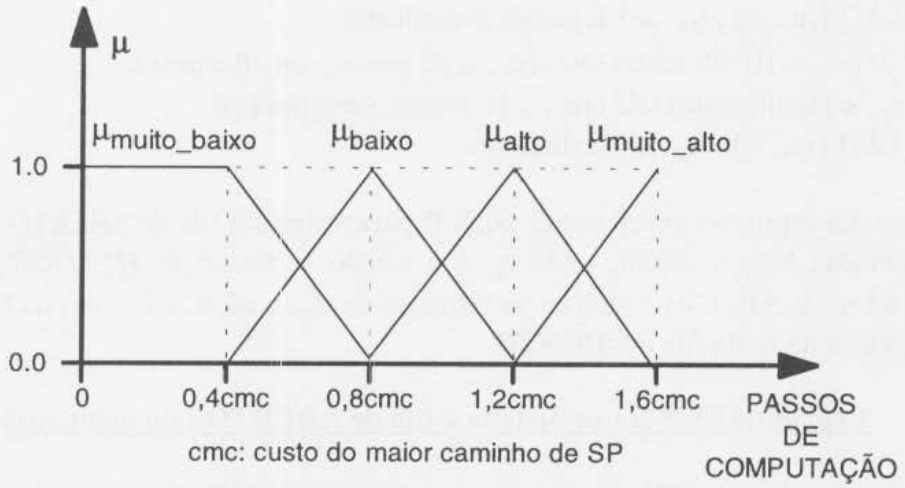

Figura 4: Funções de pertinência dos termos da variável lingüística ***custo***.

O grafo de execução do **SP** é previamente conhecido, tendo para cada um de seus nós, o registro do seu custo à raiz (***Cr***). Logo, se σ_o for o nó do **SP** equivalente ao nó objetivo η_o do **SE**, e se σ_s for o nó do **SP** equivalente ao nó η_s sucessor de *MELHOR_NÓ* (vide **A***) do **SE**, então o custo de σ_s ao nó σ_o (***Co***) é dado por:

$$Co(\sigma_s) = Cr(\sigma_o) - Cr(\sigma_s) \qquad (7)$$

O custo do nó em **SP,** equivalente ao nó sucessor para o objetivo do **SE**, deve ser ***fuzificado*** segundo o método do triângulo. Esse é então comparado às regras de inferência de um SLN para se deduzir a posição nebulosa do nó sucessor de *MELHOR_NÓ* na fila de nós abertos. Antes de serem executadas as regras do SLN que conduz à ordenação da fila de *ABERTO* (SLN B), deve-se processar um outro SLN (SLN A) que obtenha a semelhança entre os estados globais ou nós do **SE** e do **SP**. Algumas das regras desse último SLN são mostradas a seguir para o caso de duas entidades comunicantes, onde aparecem: $\mathbf{e_i}$ e $\mathbf{e_j}$, compondo um estado global do **SE**; e, e_k e e_t, compondo o estado global do **SP** equivalente ao do **SE**.

Regras do SLN A que acha a semelhança entre os estados globais do SE e do SP

⇒ SE ({e_i , e_k }É igual) E ({e_j , e_t }É igual) ENTÃO {(e_i , e_j) , (e_k , e_t)} É igual

⇒ SE (({e_i , e_k }É igual) OU ({e_i , e_k }É muito_semelhante)) E
(({e_j , e_t }É igual) OU ({e_j , e_t }É muito_semelhante))
ENTÃO {(e_i , e_j) , (e_k , e_t)} É muito_semelhante

⇒ SE ({e_i , e_k }É igual) E ({e_j , e_t }É semelhante)
ENTÃO {(e_i , e_j) , (e_k , e_t)} É muito_semelhante

⇒ SE ({e_i , e_k }É semelhante) E ({e_j , e_t }É igual)
ENTÃO {(e_i , e_j) , (e_k , e_t)} É muito_semelhante

⇒ SE ({e_i , e_k }É igual) E (({e_j , e_t }É pouco_semelhante) OU ({e_j , e_t }É diferente))
ENTÃO {(e_i , e_j) , (e_k , e_t)} É semelhante

⇒ SE (({e_i , e_k }É pouco_semelhante) OU ({e_i , e_k }É diferente)) E ({e_j , e_t }É igual)
ENTÃO {(e_i , e_j) , (e_k , e_t)} É semelhante

⇒ SE (({e_i , e_k }É semelhante) OU ({e_i , e_k }É muito_semelhante)) E
(({e_j , e_t }É semelhante) OU ({e_j , e_t }É muito_semelhante))
ENTÃO {(e_i , e_j) , (e_k , e_t)} É semelhante

⇒ SE (({e_i , e_k }É semelhante) OU ({e_i , e_k }É pouco_semelhante)) E
(({e_j , e_t }É semelhante) OU ({e_j , e_t }É pouco_semelhante))
ENTÃO {(e_i , e_j) , (e_k , e_t)} É pouco_semelhante

⇒ SE (({e_i , e_k }É diferente) OU ({e_i , e_k }É pouco_semelhante)) E
(({e_j , e_t }É diferente) OU ({e_j , e_t }É pouco_semelhante))
ENTÃO {(e_i , e_j) , (e_k , e_t)} É diferente

Algumas das regras de inferência do SLN B para ordenar a fila de *ABERTO* do algoritmo **Ã*** são apresentadas logo a seguir, onde η_s é o estado sucessor de *MELHOR_NÓ* do **SE**, equivalente ao nó σ_s do **SP**; $\tilde{C}o(\sigma_s)$ é o custo nebuloso de σ_s ao nó σ_o (objetivo no **SP**); e, $\tilde{P}(\eta_s)$ é a posição nebulosa de η_s na fila de *ABERTO*.

Regras do SLN B que ordena a fila de ABERTOS do algoritmo Ã*

⇒ SE ({η_s , σ_s}É igual) E ($\tilde{C}o(\sigma_s)$ É muito_baixo) ENTÃO $\tilde{P}(\eta_s)$ É início

⇒ SE ({η_s , σ_s}É igual) E ($\tilde{C}o(\sigma_s)$ É baixo) ENTÃO $\tilde{P}(\eta_s)$ É perto_do_início

⇒ SE ({η_s , σ_s}É igual) E ($\tilde{C}o(\sigma_s)$ É alto) ENTÃO $\tilde{P}(\eta_s)$ É meio

⇒ SE ({η_s , σ_s}É igual) E ($\tilde{C}o(\sigma_s)$ É muito_alto) ENTÃO $\tilde{P}(\eta_s)$ É fim

⇒ SE ({η_s , σ_s}É muito_semelhante) E ($\tilde{C}o(\sigma_s)$ É muito_baixo) ENTÃO $\tilde{P}(\eta_s)$ É início

⇒ SE ({η_s , σ_s}É muito_semelhante) E ($\tilde{C}o(\sigma_s)$ É baixo) ENTÃO $\tilde{P}(\eta_s)$ É perto_do_início

⇒ SE ({η_s , σ_s}É muito_semelhante) E ($\tilde{C}o(\sigma_s)$ É alto) ENTÃO $\tilde{P}(\eta_s)$ É meio

⇒ SE ({η_s , σ_s}É muito_semelhante) E ($\tilde{C}o(\sigma_s)$ É muito_alto)
ENTÃO $\tilde{P}(\eta_s)$ É perto_do_fim

⇒ SE ({η_s , σ_s}É semelhante) E ($\tilde{C}o(\sigma_s)$ É muito_baixo) ENTÃO $\tilde{P}(\eta_s)$ É perto_do_início

⇒ SE ({η_s , σ_s}É semelhante) E ($\tilde{C}o(\sigma_s)$ É baixo) ENTÃO $\tilde{P}(\eta_s)$ É meio

⇒ SE ({η_s , σ_s}É semelhante) E ($\tilde{C}o(\sigma_s)$ É alto) ENTÃO $\tilde{P}(\eta_s)$ É perto_do_fim

⇒ SE ({η_s , σ_s}É semelhante) E ($\tilde{C}o(\sigma_s)$ É muito_alto) ENTÃO $\tilde{P}(\eta_s)$ É fim

⇒ SE ({η_s , σ_s}É pouco_semelhante) E ($\tilde{C}o(\sigma_s)$ É muito_baixo) ENTÃO $\tilde{P}(\eta_s)$ É meio

$\Rightarrow$ SE ($\{\eta_s$, $\sigma_s\}$É pouco_semelhante) E ($\tilde{C}o(\sigma_s)$ É baixo) ENTÃO $\tilde{P}(\eta_s)$ É perto_do_fim

$\Rightarrow$ SE ($\{\eta_s$, $\sigma_s\}$É pouco_ semelhante) E ($\tilde{C}o(\sigma_s)$ É alto) ENTÃO $\tilde{P}(\eta_s)$ É fim

$\Rightarrow$ SE ($\{\eta_s$, $\sigma_s\}$É pouco_ semelhante) E ($\tilde{C}o(\sigma_s)$ É muito_alto) ENTÃO $\tilde{P}(\eta_s)$ É fim

$\Rightarrow$ SE $\{\eta_s$, $\sigma_s\}$É diferente ENTÃO $\tilde{P}(\eta_s)$ É fim

Essa técnica está sendo usada na implementação de um verificador, cuja arquitetura pode ser vista na figura 5. Tal verificador é um aperfeiçoamento de um outro [15], desenvolvido para um Sistema de Auxílio ao Projeto de Protocolos de Comunicação.

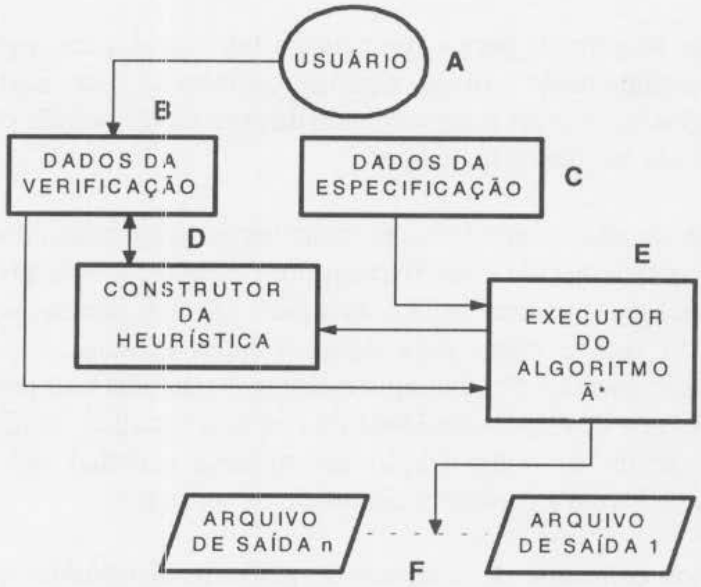

Figura 5: Arquitetura do Verificador.

O bloco ***A*** da figura 5 representa o usuário que interage com o sistema por meio da definição de dados gerais para a verificação (bloco ***B***). A heurística a ser empregada no auxílio da validação é criada pelo bloco ***D*** com a utilização dos Sistemas de Lógica Nebulosa (SLN A e SLN B). O bloco ***E*** consiste no motor do sistema proposto, e tem a finalidade de validar uma dada propriedade, a partir da consulta aos dados da verificação e da especificação (bloco ***C***). Por fim, o bloco ***F*** compõe os arquivos de saída que relatam a situação da verificação. Dos módulos apresentados, apenas o bloco ***E*** encontra-se em fase final de construção.

5 GERÊNCIA DE GRANDES REDES DE TELECOMUNICAÇÕES

O principal problema de uma grande rede é não poder ser gerida apenas por um gerente. Mesmo que seja admitido o aumento da capacidade dos novos equipamentos, sempre teremos um limite, ora por causa do processamento, ora por causa da velocidade do canal de comunicação. Assim, o sistema deve ser flexível, escalonável, podendo a sua capacidade ser aumentada de acordo com as necessidades.

A solução seria implementar um sistema distribuído, com vários gerentes, e que cada um deles pudesse gerir um grupo de agentes. Mas é importante atentar para o fato de que a existência de vários gerentes pode provocar um problema de autoridade. Por isso, cada um deve saber o que pode controlar e a quem deve se reportar, de forma que o sistema fique coerente e não haja conflito de instruções entre gerentes diferentes. Em um sistema grande não se pode também

vincular rigidamente um agente a um gerente, pois, no caso de um gerente ficar inoperante, deve-se ter a possibilidade de outro gerente assumir o controle.

Uma arquitetura hierárquica é a mais adequada nesse caso, mas ela não deve ser rígida no que concerne às relações entre agentes e gerentes, nem provocar muito retardo na transmissão de informações (por exemplo, uma medida corretiva deve ser tomada pelo gerente imediatamente superior e não deve demorar muito para chegar até ele).

O sistema deve ser flexível também quanto à possibilidade de acrescentar elementos geridos dinamicamente, a medida que sejam ativados, assim como retirá-los caso sejam desligados.

Uma informação importante para a gerência é a hora local, para registrar, por exemplo, o momento em que o equipamento sofreu alguma ocorrência. Por isso, é necessário uma sincronização dos relógios no sistema e, dependendo da precisão requerida para ele, a conferência da hora local deve ser mais ou menos freqüente.

Um sistema hierárquico deve também respeitar a responsabilidade e autoridade das empresas que operam os sistemas de telecomunicação. Como é comum haver diversas empresas operando as telecomunicações em uma região, uma para ligações dentro do mesmo estado, outra para ligações dentro do país e outra para ligações internacionais, é necessário respeitar a autoridade sobre a área de atuação. Por exemplo a empresa nacional não pode controlar um canal dentro do estado, pois esse é de responsabilidade da empresa estadual. Porém a empresa nacional pode regras para os canais de comunicação da empresa estadual que são utilizados para comunicação nacional, de forma a garantir a qualidade de serviço.

Dentre os vários conceitos de sistemas de gerência distribuída, que podem ser vistos melhor em [7], convém destacar o conceito de diretriz de gerência. A gerência de sistemas distribuídos envolve as funções de monitorar as atividades, tomar decisões sobre gerência e executar ações de controle para modificar o comportamento do sistema. Pela própria filosofia desse sistema, o gerente deve ter autonomia para decidir as ações que devem ser tomadas, sem precisar receber uma ordem explícita de um gerente superior. Essas decisões, no entanto, dentro do sistema distribuído, com vários objetos executando a função de gerência, precisam de uma diretriz que oriente sua realização.

Diretriz é uma informação que influencia o comportamento de um objeto, isso é, que diz como deve ser a sua interação com os demais objetos do sistema. Toda diretriz tem um sujeito, no qual o objetivo deve atuar, e um alvo, que é o comportamento desejado. Em alguns casos o alvo ou o sujeito poderão ser implícitos, como, por exemplo, uma diretriz que determina que toda senha deve ter mais de oito caracteres, tem como sujeito implícito qualquer usuário que cria ou troca senha.

6 VERIFICAÇÃO DE UM SISTEMA DE GERÊNCIA HIERÁRQUICA

O Sistema de Gerência Hierárquica (SGH) a ser verificado, proposto em [7], é mostrado esquematicamente na figura 6. Todos os módulos do SGH estão localizados na camada Aplicação do modelo OSI. As camadas inferiores, Física até Apresentação, estão representadas genericamente na figura 6 como Meio de Comunicação. Acima delas está a camada que oferece o serviço CMISE (*Common Management Information Service Element*), e que se comunica com

outros dispositivos através do protocolo CMIP (*Common Management Information Protocol*) [16].

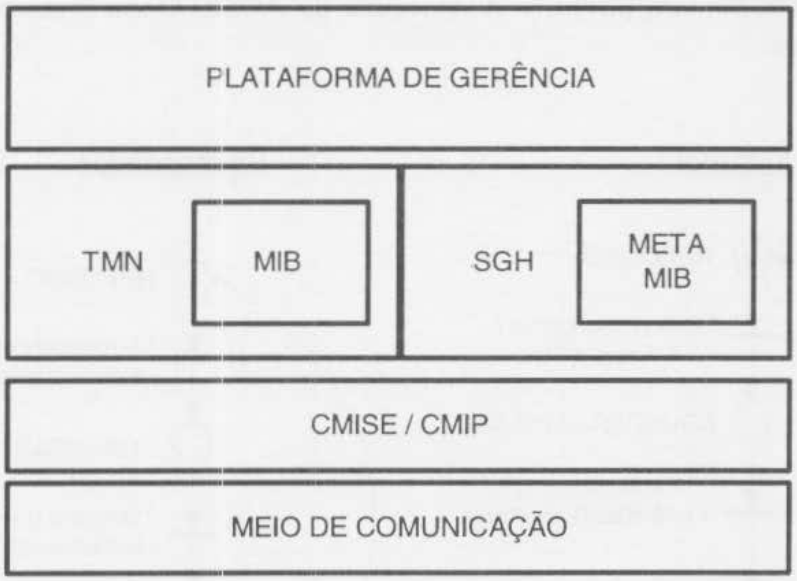

Figura 6: Arquitetura do sistema de gerência.

O módulo TMN (*Telecommunication Management Network*) executa as funções de coleta de informações dos gerentes e as consolida em suas tabelas de dados de gerência (MIBs - *Management Information Base*). Esse módulo implementa o comportamento de gerente e atende às necessidades de gerência de uma rede de telecomunicações.

A camada de nível mais elevada foi chamada de Plataforma de Gerência e consiste em um sistema em ambiente gráfico, que oferece as informações de gerência para o operador de redes.

O módulo SGH executa as funções de gerência de diretrizes e gerência de delegação. Ele assume o comportamento de agente, fornecendo informações de gerência do gerente um nível acima, utilizando os dados consolidados na Meta-MIB. A Meta-MIB apresenta para o gerente acima a imagem da rede gerida, segundo a visão do nível superior. O SGH é responsável também por ler as informações das MIBs TMN e convertê-las para a Meta-MIB.

A tabela 1 mostra as primitivas do SGH e a figura 7 apresenta o modelo da primitiva **PolicyAChange** especificado por rede de predicado-ação. Para facilitar o entendimento, neste artigo, será verificada apenas a especificação da primitiva **PolicyAChange**, responsável pela alteração das diretrizes de um agente ou gerente filho.

Tabela 1: Descrição das primitivas de serviço do SGH.

PRIMITIVA	DESCRIÇÃO	PRIMITIVA	DESCRIÇÃO
ManagerInit	Inicia um gerente no sistema	DelegAct	Passa gerência para outro
ManagerRead	Lê a tabela de gerentes	PolicyMRead	Lê tabela de diretrizes do gerente
AgentRead	Lê a tabela de agentes	PolicyARead	Lê tabela de diretrizes do agente
AgentElimin	Retira agente da tabela de agentes	PolicyAChange	Altera diretrizes do agente
AgentCreate	Inclui agente na tabela de agentes	PolicyMLoad	Atualiza diretrizes locais
DelegAsk	Pede autorização para delegar		

A primitiva **PolicyAChange** é implementada com um ciclo de mensagem **M-Set** para enviar uma alteração na tabela de diretrizes (**TablePolicy**). Inicialmente é realizada a associação da comunicação, utilizando a primitiva **A-Associate** do ACSE. O encerramento é feito com a primitiva **A-Release**.

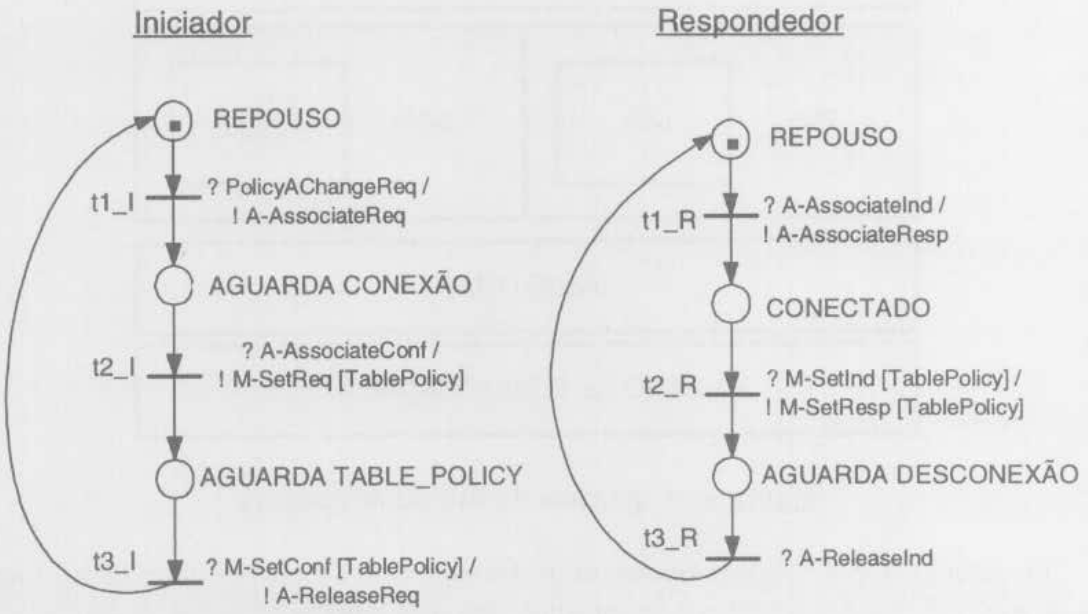

Figura 7: Modelo da primitiva PolicyAChange.

Uma vez que foi introduzido o **SE**, como sendo o protocolo SGH, agora pode ser observado na figura 8, a rede de predicado-ação que modela uma entidade com o comportamento do serviço de abertura de conexão, seguida de transmissão de dados que caracteriza o **SP**. Na figura 9, pode-se ver o modelo do meio de comunicação para o **SP**.

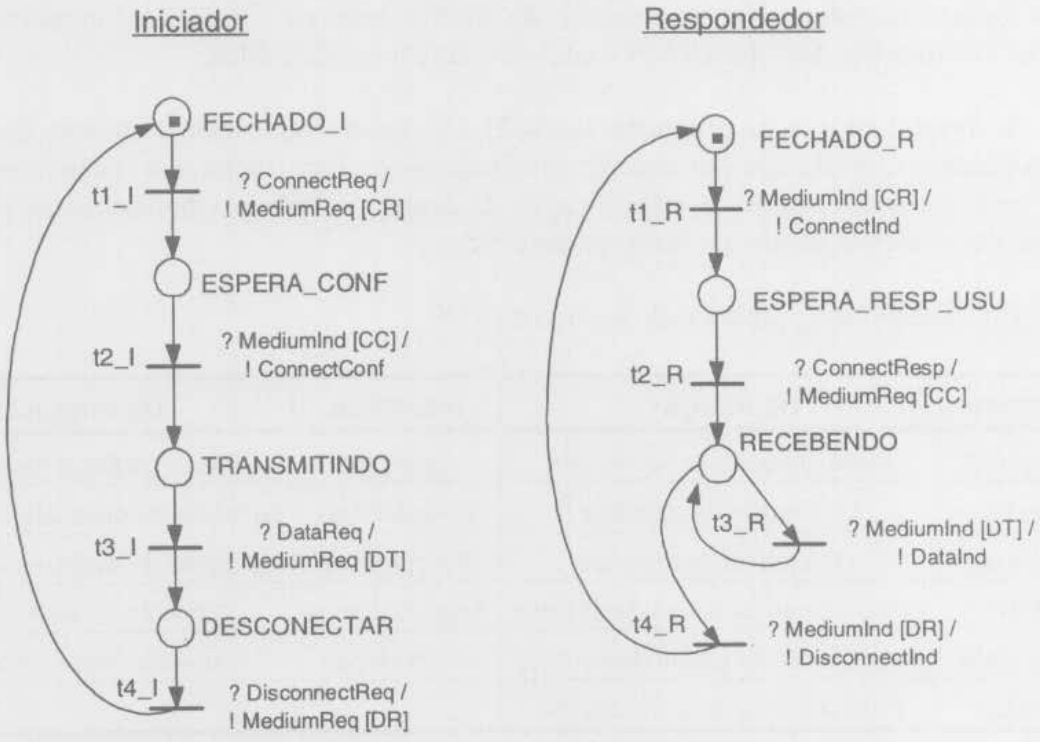

Figura 8: Modelo dos comportamentos Iniciador e Respondedor do **SP**.

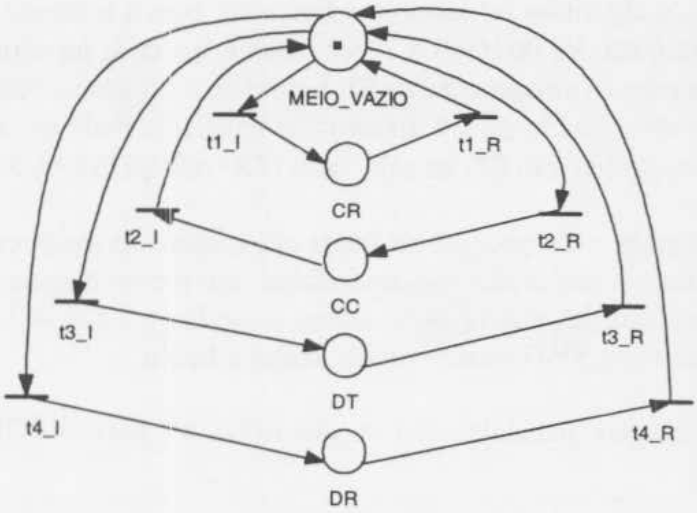

Figura 9: Modelo do meio de comunicação (*Medium*) do **SP**.

Aplicando-se agora a técnica de verificação, devemos inicialmente estabelecer a relação nebulosa $\tilde{R}_{E\mathcal{E}}(e,\epsilon)$ e um estado global objetivo η_o. Antes disso, convém destacar que os estados parciais *FECHADO_I* e *FECHADO_R* do **SP** são os mesmos no caso de suas entidades apresentarem os dois comportamentos para a conexão. No que diz respeito ao **SE**, o exemplo a ser analisado será o de duas entidades (G1 e G2) comunicantes com os dois comportamentos (simultâneo) mostrados na figura 7, de tal forma que um gerente **G1** possa enviar uma mudança de diretriz para um outro gerente **G2**, assim como, **G2** pode enviar também uma mudança de diretriz para um de seus agentes, além de responder a **G1**.

Sejam:

E = {*REPOUSO, AGUARDA CONEXÃO, AGUARDA TABLE POLICY, CONECTADO, AGUARDA DESCONEXÃO*};
$\mathcal{E}$ = {*FECHADO, ESPERA_CONF, ESPERA_RESP_USU, TRANSMITINDO, DESCONECTAR, RECEBENDO*};

$$\tilde{R}_{E\mathcal{E}}(e,\epsilon) = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.9 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.9 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.7 \end{bmatrix};$$

η_o = (*AGUARDA TABLE POLICY, AGUARDA DESCONEXÃO*).

Na matriz $\tilde{R}_{E\mathcal{E}}(e,\epsilon)$, as linhas caracterizam os elementos de **E** na ordem em que se vê acima, enquanto as colunas são os elementos de $\mathcal{E}$, também na mesma ordem em que se vê acima. Logo, cada elemento de $\tilde{R}_{E\mathcal{E}}(e,\epsilon)$ é o grau de equivalência entre um elemento de **E** e um elemento de $\mathcal{E}$, por exemplo, o valor **0.9** na terceira linha e quarta coluna é a equivalência ou semelhança entre *AGUARDA TABLE POLICY* e *TRANSMITINDO*. O estado global objetivo η_o dado caracteriza uma mudança de diretriz enviada do gerente **G1** para o gerente **G2**.

Fazendo-se então a ***fuzificação*** pelo método simples dos elementos de **E**; procedendo com a composição de relações dada pela equação (6); e, logo em seguida, obtendo-se a ***desfuzificação*** de $\tilde{O}(\epsilon)$, chega-se a conclusão que σ_o = (*TRANSMITINDO, RECEBENDO*).

De posse de σ_o, o algoritmo **A*** começa a funcionar com o primeiro nó da fila *ABERTO*, que é a raiz η_r = (*REPOUSO, REPOUSO*) e, o seu custo à η_o, deve ser estimado a partir de **SP**. Na tabela 2 é fornecido o grafo de execução de **SP** feito com o programa ARP [17], onde pode-se observar, que o estado **M4** é obtido após **4** disparos de transição. Pode-se ver, na tabela 2, que o estado **M4** caracteriza o objetivo em **SP**, ou seja, σ_o = (*TRANSMITINDO, RECEBENDO*).

No grafo de **SP** (tabela 2), pode-se ver ainda que o custo do maior caminho é **7** ao atingir o estado **M7**, pois as demais marcações são alcançadas com menos disparos de transição. Logo, o valor máximo para o eixo das abscissas na figura 4, seria (1.6 x 7 = 11.2) e, nesse caso, o custo **4** da raiz ao objetivo em **SP** fica entre **muito_baixo** e **baixo**.

O início da execução (manualmente) do algoritmo **A*** para o SGH é apresentado nas tabelas 3, 4 e 5 a seguir.

Tabela 2: Grafo de acessibilidade da rede do SP.

Marcação ou Estado Global	Transições Disparadas	Identificação do Estado Global
M0	(t1_I: M1)	{Fechado_I, Fechado_R, MeioVazio}
M1	(t1_R: M2)	{Fechado_R, EsperaConf, CR}
M2	(t2_R: M3)	{MeioVazio, EsperaConf, EsperaRespUsu}
M3	(t2_I: M4)	{EsperaConf, Recebendo, CC}
M4	(t3_I: M5)	{MeioVazio, Transmitindo, Recebendo}
M5	(t3_R: M6)	{Desconectar, Recebendo, DT}
M6	(t4_I: M7)	{MeioVazio, Desconectar, Recebendo}
M7	(t4_R: M0)	{Fechado_I, Recebendo, DR}

Tabela 3: Valores dos principais dados do Algoritmo A* em execução.

Executando A* (passo 1)			
ABERTO	0 (raiz)	nil	
FECHADO	nil		
MELHOR_NÓ:	SUCESSORES:		
Executando A* (passo 2)			
ABERTO	nil		
FECHADO	0 (raiz)	nil	
MELHOR_NÓ: 0 (raiz)	SUCESSORES: 100		
Executando Trata_Sucessores			
Parâmetro Suc	1		
VELHO	nil		
VELHO	nil		
Executando Colocar_ordenado_em_aberto			
Parâmetro Suc	1		
ABERTO	1	nil	
FECHADO	0 (raiz)	nil	
Executando A* (passo 3)			
ABERTO	1	nil	
FECHADO	0 (raiz)	nil	
MELHOR_NÓ: 0 (raiz)	SUCESSORES: 100		
Executando Trata_Sucessores			
Parâmetro Suc	2		
VELHO	nil		
VELHO	nil		
Executando Colocar_ordenado_em_aberto			
Parâmetro Suc	2		
ABERTO	2	1	nil
FECHADO	0 (raiz)	nil	

A tabela 3 mostra um trecho da execução do algoritmo **A***, onde se tem inicialmente, no primeiro passo, o nó raiz na fila *ABERTO*, em seguida, no passo **2**, o nó raiz vai para a fila *FECHADO*, quando então são criados os seus sucessores (nós 1 e 2). Nesse segundo passo, é chamado o procedimento **Trata_Sucessores**, que tem como função atualizar o valor de **g** (custo do nó desde a raiz), substituindo, caso o nó já tenha sido obtido anteriormente, o novo valor de **g** calculado e o nó pai do nó em tratamento, pelos valores antigos já armazenados.

O procedimento **Trata_Sucessores** chama a rotina **Colocar_ordenado_em_aberto** que, ao usar o SLN A e o SLN B, providencia a alocação dos nós na fila *ABERTO*. No segundo passo do **A*** (primeira vez em que Colocar_ordenado_em_aberto é executado) o sucessor **1** fica sozinho na fila *ABERTO*.

Tabela 4: Valores da estrutura de dado NÓ da execução do Algoritmo A*.

NÓ	DADOS	g	h'	ptrSuc	ptrPai	próximo
0	REPOUSO, REPOUSO	0		nil	nil	nil
1	AGUARDA CONEXÃO, REPOUSO	1			0	nil
2	REPOUSO, AGUARDA CONEXÃO	1			0	
0	REPOUSO, REPOUSO			102		

A tabela 4 mostra a evolução dos valores dos campos dos nós na memória. Inicialmente, o nó **0** (raiz) criado, depois são gerados os seus sucessores (nós 1 e 2), em seguida, um ponteiro para a fila de sucessores da raiz é armazenado no seu campo **ptrSuc**.

Tabela 5: Valores da estrutura de dado NÓ_SUC da execução do Algoritmo A*.

NÓ_SUC	próximo	pSuc	NÓ_SUC	próximo	pSuc
100	101	1	102	103	1
101	nil	2	103	nil	2
102	nil	1			

A tabela 5 mostra a evolução das estruturas de nós sucessores criadas na memória identificadas pelos endereços arbitrários **100,101** etc. Inicialmente, pode-se observar os nós sucessores da raiz **100** e **101** gerados a partir da função **GerarSuc** no passo 2 do **A***, em seguida na primeira execução de **Trata_Sucessores**, o nó **102** é criado e registrado como o primeiro sucessor da raiz (tabela 4). Na segunda execução de **Trata_Sucessores**, é criado o nó **103** e é enfileirado após o nó **102**.

O procedimento **Colocar_ordenado_em_aberto** do passo 3 da execução do **A*** (tabela 3) primeiro calcula os estados parciais equivalentes a *AGUARDA CONEXÃO* e *REPOUSO* pela equação (6), ou seja:

$$\textit{REPOUSO fuzificado} = [1\ 0\ 0\ 0\ 0],\ \text{então}$$
$$[1\ 0\ 0\ 0\ 0] \circ \tilde{R}_{E\mathcal{E}}(e,\epsilon) = [1\ 0\ 0\ 0\ 0\ 0] = \textit{FECHADO fuzificado}$$

$$\textit{AGUARDA CONEXÃO fuzificado} = [0\ 1\ 0\ 0\ 0],\ \text{então}$$
$$[0\ 1\ 0\ 0\ 0] \circ \tilde{R}_{E\mathcal{E}}(e,\epsilon) = [0\ 1\ 0\ 0\ 0\ 0] = \textit{ESPERA\_CONF fuzificado}$$

Logo, o estado global em **SP** equivalente ao nó **2** no **SE** é σ_2 = *(FECHADO, ESPERA_CONF)* que equivale à marcação M1 do grafo de **SP**, conforme mostra a tabela 2.

Após calcular σ_2, **Colocar_ordenado_em_aberto** faz a sua medida de semelhança com o nó **2** de **SE**, ao executar o SLN A, obtendo como resultado que σ_2 e o nó **2** de **SE** são **iguais**. Em

seguida, é executado o SLN B, indicando que o nó **2** deve ser colocado no **início**, pois o custo de σ_2 é considerado **muito_baixo** (ver figura 4 e primeira regra de SLN B). Com isso, o nó **2** de **SE** fica na primeira posição da fila *ABERTO* no passo 3 do algoritmo **A*** (tabela 3). Na execução do sétimo passo do algoritmo **A***, o nó objetivo é alcançado.

7 COMENTÁRIOS FINAIS

A comparação de um sistema sob estudo (SE) com um sistema padrão (SP), livra o usuário de codificar o seu conhecimento a respeito do projeto de protocolos de tal forma que um algoritmo de busca informado, como o **A***, pode ser implementado de maneira mais automática. O emprego da Lógica Nebulosa na formalização dessa comparação, entre o **SE** e o **SP**, mostrou-se promissor pelo fato de a mesma dispor de recursos adequados e conhecidos para lidar com aproximações, essência da técnica de verificação.

A técnica de verificação, aqui apresentada, faz uma comparação entre o **SE** e o **SP**, levando-se em consideração apenas os estados parciais das suas respectivas especificações. No sentido de aperfeiçoar ainda mais o desempenho da técnica, é possível reescrever as regras de inferência do Sistema de Lógica Nebulosa que ordena a fila de *ABERTO* (SLN B), incorporando a manipulação da medida de semelhança entre outros objetos das especificações do **SE** e do **SP** como, por exemplo, as transições, as variáveis locais, e as mensagens nas filas.

Além da incorporação de novos objetos para comparar os dois sistemas (**SE** e **SP**), pode-se também construir a técnica com uma comparação de prefixos de computação ou trechos de caminhos dos grafos de **SE** e **SP**. Nesse sentido, seria possível averiguar melhor o grau de semelhança entre os dois sistemas.

A técnica de verificação demonstrou, no exemplo apresentado, ser viável, uma vez que o objetivo é logo atingido. Entretanto, ficou evidente que o grande sucesso da técnica fica na dependência de um sistema padrão (SP) que se assemelhe ao sistema sob estudo (SE), caso contrário, o tempo gasto no processamento dos Sistemas de Lógica Nebulosa será inútil. Logo, a técnica de verificação, no momento, tem uma utilização limitada, somente para aqueles sistemas parecidos com os padrões codificados.

No sentido de contornar a questão dos sistemas sob estudo muito diferentes dos sistemas padrão codificados, será feito um ensaio para averiguar a eficácia em se comparar partes de um sistema sob estudo com vários sistemas padrão mais simples, de tal forma a se obter um tipo de parametrização do sistema sob estudo.

8 REFERÊNCIAS BIBLIOGRÁFICAS

[1] GOUDA, M. G. e HAN, J. Y., *Protocol validation by fair progress state exploration*, Computer Networks & ISDN System, 9, 1985.

[2] ZHAO, Z. R. e BOCHMANN, G. V., *Reduced reatchability analysis of communication protocols: A new approach*, Technical report, University Montreal, 1986.

[3] HSIEH, W. S., NAIN, T. S., YANG, M. S., LU, C. S., HUANG, K. C. e TSENG, J. R., *A Fast Method of Protocol Validation Using Reduced Stable State Exploration Technique*, Microprocessing and Microprogramming, North-Holland, 1991.

[4] JARD, C., *Valider les protocoles en simulant*, Colloque Francophone Sur L'Ingénierie des Protocoles - CFIP'88, Bordeaux, setembro, 1988.

[5] PAGEOT, J. M., *Une expérience de simulation guidée*, Colloque Francophone Sur L'Ingénierie des Protocoles - CFIP'88, Bordeaux, setembro, 1988.

[6] WEST, C. H., *Protocol validation - principles and applications*, Computer Networks and ISDN Systems, North-Holland, 1992.

[7] FERNANDEZ, M. P. e PEDROZA, A. C. P., *Especificação de um Protocolo para Gerenciamento Hierárquico em Ambiente TMN*, XV Simpósio Brasileiro de Redes de Computadores, 1997.

[8] FERNANDEZ, M. P. e PEDROZA, A. C. P., *Sistema de Gerenciamento Hierárquico para Grandes Redes de Telecomunicações*, XV Simpósio Brasileiro de Telecomunicações, 1997.

[9] ZIMMERMANN, H. J., *Fuzzy Set Theory - and Its Applications*, Kluwer-Nijhoff Publishing, 1985.

[10] KANDEL, A., *Fuzzy Mathematical Techniques with Applications*, Addison-Wesley Publishing Co., 1986.

[11] MENDEL, J. M., *Fuzzy Logic Systems for Engineering: A Tutorial*, Proceedings of the IEEE, vol. 83, nº 3, março, 1995.

[12] ZADEH, L. A., *Fuzzy Sets*, Information and Control, 1965.

[13] SILVA, A. P. da, *Escalamento Heurístico Baseado em Lógica Difusa*, Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1995.

[14] RICH, E. e KNIGHT, K., *Artificial Intelligence*, Second Edition, International Edition, McGraw-Hill Inc., 1991.

[15] BERNARDO FILHO, O., PEDROZA, A. C. P. e LEÃO, J. L. S., *O Verificador de Um Sistema de Auxílio ao Projeto de Protocolos*, Anais do 9º Congresso Brasileiro de Automática, setembro, 1992.

[16] LEHMAN Jr., E. e PEDROZA, A. C. P., *Especificação e verificação do protocolo CMIP para gerenciamento de redes*, XII Simpósio Brasileiro de Redes de Computadores, 1994.

[17] ARP 2.3, Analisador de Redes de Petri, Tese de Mestrado, LCMI-EEL-UFSC.