UMA FERRAMENTA PARA VERIFICAÇÃO DE SISTEMAS DISTRIBUÍDOS COM LÓGICA NEBULOSA: IMPLEMENTAÇÃO E EXPERIÊNCIAS

Orlando Bernardo Filho[1,2], Aloysio C. P. Pedroza[2,3] e Jorge L. S. Leão[2]

(1) Departamento de Enga. de Sistemas e Computação/FEN-UERJ

Rua São Francisco Xavier nº 524, bloco D, 5º andar, sala 15, CEP 20550-013

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/aloysio@gta.ufrj.br/leao@gta.ufrj.br

(3) Departamento de Eletrônica/EE-UFRJ

Resumo: Este trabalho apresenta uma pesquisa realizada sobre técnicas formais de especificação e verificação que permitiu desenvolver uma ferramenta capaz de verificar propriedades de sistemas distribuídos. Esta ferramenta, denominada **Verest+**, baseia-se na Lógica Nebulosa e na Lógica Temporal. São discutidas experiências com um protocolo para a gerência de redes de computadores, com o protocolo Abracadabra, padronizado pela ISO (***International Organization for Standardization***), e com um protocolo multiponto-multiponto para texto-conferência.

Palavras Chaves: Lógica Nebulosa, Heurística, Verificação, Protocolos, Modelos de Espaço de Estados.

Abstract: This paper presents a research on formal specification and verification techniques. The results obtained allowed the development of a verification tool capabable of verifying distributed systems properties. The tool, called **Verest+**, is based on Fuzzy and Temporal logics. This paper also discusses experiments with a protocol for computer networks management, the ISO (International Organization for Standardization) Abracadabra protocol and a multipoint-multipoint text conference protocol.

Keywords: Fuzzy Logic, Heuristics, Verification, Protocols, State Space Models.

1 INTRODUÇÃO

Os métodos formais têm sido usados no projeto de redes de computadores desde a década de 70. Embora algumas ferramentas criadas tenham atingido um bom nível de automação, ainda se percebe uma grande dificuldade para o projetista usá-las, uma vez que a complexidade dos métodos envolvidos é grande. Os métodos formais continuaram se desenvolvendo nessa década de 90 e, atualmente, existe uma preocupação em se propor técnicas e construir ferramentas que sejam de fácil utilização, além de fornecerem respostas mais rápidas.

O trabalho aqui descrito procurou se guiar por esses princípios atuais, uma vez que a ferramenta para verificação de protocolos de comunicação criada (**Verest+**) emprega um Sistema de Lógica Nebulosa que otimiza a busca no grafo de execução do protocolo sob análise, acelerando a obtenção dos objetivos (Bernardo Filho, 1999).

Este artigo encontra-se dividido em cinco seções. Na seção seguinte, são introduzidos os principais conceitos da Lógica Temporal, da Lógica Nebulosa e da técnica de busca heurística empregadas na construção da ferramenta de verificação **Verest+**. Na terceira seção, mostra-se a técnica de verificação criada. A seção 4 mostra a implementação do **Verest+** onde podem ser vistas as descrições de todos os seus módulos. Apresenta-se, na quinta seção, os resultados das experiências realizadas com os protocolos de gerência hierárquica de rede, Abracadabra e Multiponto-Multiponto.

2 FUNDAMENTOS TEÓRICOS

A ferramenta de verificação **Verest+** foi construída aplicando-se os conceitos das lógicas nebulosa e temporal, além da técnica de busca heurística. Os protocolos a serem testados pelo **Verest+** devem ser especificados em ESTELLE (ISO, 1989).

2.1 Lógica Temporal

O **Verest+** usa as fórmulas da Lógica Temporal para descrever as propriedades a serem testadas. A Lógica Temporal (Rescher e Urquhart, 1971; Manna e Pnueli, 1981) é um ramo especial da lógica que lida com o desenvolvimento de situações no tempo. Enquanto a lógica ordinária é adequada para descrever uma situação estática, a Lógica Temporal permite discutir como uma situação muda devido à passagem do tempo.

São definidos alguns operadores para a formação das fórmulas da Lógica Temporal. Aqui são apresentados apenas os três operadores usados no **Verest+**.

Artigo Submetido em 15/07/99
1a. Revisão em 04/10/99; 2a. Revisão em 05/09/00; 3a. Revisão em 02/03/01;
Aceito sob recomendação do Eda. Consultora Profa. Dra. Sandra A. Sandri

- Operador ***sempre*** (□) cuja fórmula formada com uma condição φ (□φ), é verdadeira em um estado s_i, se e somente se, φ for verdade em todos os estados s_j acessíveis a partir de s_i ($j \geq i$).

- Operador ***eventualmente*** (◊) que ao ser aplicado a uma condição φ (◊φ), é verdade no estado s_i, se e somente se, φ for verdade em algum estado acessível de s_i (podendo ser no próprio s_i).

- Operador ***até*** (***u***) onde uma fórmula básica com duas condições φ e ψ (φ ***u*** ψ) assume valor verdade a partir de s_i, se e somente se, ψ for verdade em algum estado acessível de s_i (incluindo s_i) e φ se mantém verdade, enquanto ψ não for. Não é necessário que φ seja verdade no exato estado no qual ψ o seja.

Por questões de simplicidade de implementação do verificador desenvolvido, limitou-se a quantidade de fórmulas a ser utilizadas a apenas um conjunto pré-definido. A variação das fórmulas resume-se aos operandos das mesmas, ou seja, uma dada fórmula pode ser empregada com proposições distintas de tal forma a descrever propriedades distintas dos protocolos.

Algumas das fórmulas da lógica temporal do conjunto pré-definido são apresentadas a seguir, onde podem ser vistos os estados globais operandos ***S1***, ***S2*** e ***S3*** como proposições das fórmulas. Tais estados representam um contexto da computação do protocolo sob estudo.

- *S1* ***u*** *S2*
- *S1* ⊃ (*S2* ***u*** *S3*)
- □(*S1* ***u*** *S2*)
- ◊(*S1* ***u*** *S2*)

No conjunto de fórmulas pré-definido, o símbolo ⊃ denota uma implicação lógica cuja semântica é derivada dos conectivos ~ (***não***) e ∨ (***ou***). Tal semântica é da forma a seguir, onde **P** e **Q** são proposições.

P ⊃ **Q** é verdade se e somente se ~ **P** ∨ **Q** é verdade (1)

O conjunto pré-definido das fórmulas da lógica temporal foi escolhido com base nas propriedades mais comuns que normalmente são testadas nos sistemas concorrentes, segundo uma classificação proposta por Manna e Pnueli (1981). De acordo com essa classificação, as propriedades dos sistemas concorrentes podem ser agrupadas em três categorias principais: 1) ***segurança*** ou ***invariância***, 2) ***vivacidade*** ou ***eventualidade*** e 3) ***precedência***.

As propriedades de segurança são normalmente expressas por fórmulas do tipo:

$$\psi \qquad ou \qquad \varphi \supset \psi \tag{2}$$

Os termos ψ e φ são fórmulas que não envolvem operadores temporais, expressando condições sobre alguns objetos da especificação de um protocolo que, no caso da linguagem ESTELLE, poderia ser: estados de módulo (entidades ou processos), variáveis de corpo ou mensagens. As principais propriedades de segurança são: correção parcial, invariância global e local, exclusão mútua e ausência de bloqueio.

As propriedades de vivacidade podem ser expressas por fórmulas do tipo:

$$\Diamond\psi \qquad ou \qquad \varphi \supset \Diamond\psi \tag{3}$$

Nesse grupo podem-se encontrar as propriedades de correção total, asserções intermitentes, acessibilidade e progressão.

As propriedades de precedência são normalmente representadas por fórmulas do tipo:

$$\varphi \; u \; \psi \qquad ou \qquad (\varphi \; u \; \psi) \tag{4}$$

As duas mais importantes propriedades de precedência são: vivacidade segura e ausência de resposta não solicitada.

A técnica de verificação implementada, embora teste propriedades de segurança, não foi idealizada com esse propósito, já que a heurística empregada baseia-se estritamente em buscas parciais no grafo de execução do protocolo, o que proporciona, naturalmente, a análise das propriedades de vivacidade e precedência, pois essas referem-se a algum caminho da árvore do protocolo e não a todos os caminhos (busca exaustiva).

A tabela 1 a seguir mostra as fórmulas usadas pelo **Verest+** na forma como deve ser feita a entrada do seu arquivo de definição dos dados das fórmulas de lógica temporal. Uma vez que tal arquivo é de formato ASCII, para ser possível editar as fórmulas usadas, foi necessário trocar os símbolos □, ◊ e ⊃ para [] (dois colchetes), & e ->, respectivamente.

Tabela 1 – Fórmulas da lógica temporal que podem ser usadas na análise de um sistema distribuído. As proposições *S1*, *S2* e *S3* são estados globais do protocolo sob teste.

Nº	Fórmula	Descrição
1	S1 u S2	S1 até S2
2	S1 -> (S2 u S3)	S1 implica (S2 até S3)
3	[](S1 u S2)	Sempre (S1 até S2)
4	&(S1 u S2)	Eventualmente (S1 até S2)

2.2 Lógica Nebulosa

Na matemática clássica, um conjunto é definido como uma coleção de elementos distintos ou objetos que pode ser finita ou não. Tal conjunto pode ser descrito de várias maneiras, como, por exemplo, enumerando cada um de seus elementos ($A = \{1,2,3,4\}$), ou então, a partir de uma condição de pertinência ($A = \{x \in \mathbf{N} \mid x<5\}$).

Ao usar a descrição a partir de uma condição de pertinência, se um elemento ***x*** causar a avaliação dessa condição como verdadeira, então ele pertence ao conjunto; se a avaliação for falsa, então ele 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** em caso contrário.

Na teoria dos conjuntos nebulosos (Zimmermann, 1985; Mendel, 1995; Zadeh, 1965), a função de pertinência não retornará apenas os valores **0** ou **1**, mas qualquer outro valor do intervalo **[0,1]**, o que significa que pode haver vários graus de pertinência. Sendo assim, um conjunto nebuloso $\tilde{A}$ possui o seguinte aspecto:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\} \tag{5}$$

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 ***crisp*** (abrupto ou preciso).

Olhando a definição dos conjuntos nebulosos, esses podem ser vistos como um conjunto clássico (***crisp***) de pares. O exemplo abaixo mostra o conjunto nebuloso $\tilde{A}$ dos números inteiros próximos de 4.

$$\tilde{A} = \{(1,0.3), (2,0.6), (3,0.8), (4,1.0), (5,0.8), (9,0.6), ...\} \quad (6)$$

Como se observa, **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.

$$\tilde{A}=\{(x,\mu_{\tilde{A}}(x))|\,\mu_{\tilde{A}}(x)=(1+(x-4)^2)^{-1}\} \quad (7)$$

A Teoria dos Conjuntos Nebulosos tem vários sub-ramos, entre eles, a Aritmética Nebulosa, a Programação Matemática Nebulosa, a Teoria de Grafos Nebulosos e a Lógica Nebulosa. É comum empregar o termo Lógica Nebulosa tanto no seu sentido mais restrito, referindo-se à lógica do raciocínio aproximado, como num sentido mais amplo, praticamente como sinônimo de Teoria dos Conjuntos Nebulosos (Zadeh, 1965).

As relações nebulosas são subconjuntos nebulosos do produto cartesiano X x 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

$$\tilde{R}=\{((x,y),\mu_{\tilde{R}}(x,y))|(x,y)\in X\times Y\} \quad (8)$$

é chamada de uma relação nebulosa em X x Y.

Como exemplo, pode-se citar a relação nebulosa ***x bem menor que y***, cuja função de pertinência é mostrada na figura 1.

y \ *x*	1	2	3	4
0.1	0	0	0	0
1	0	0	0	0
10	0.9	0.8	0.7	0.6
100	1	1	1	1

Figura 1 – Representação matricial da relação nebulosa *x bem menor que y*.

$$\mu_{\tilde{R}}(x,y)=\begin{cases}0 & para \quad y\le x\\ \dfrac{y-x}{10y} & para \quad x<y\le 11x\\ 1 & para \quad y>11x\end{cases} \quad (9)$$

Na definição acima, foi feito um mapeamento de X x Y em **[0,1]**; entretanto pode-se também obter uma relação nebulosa a partir de dois outros conjuntos nebulosos, como é apresentado a seguir.

Sejam $X,Y \subseteq R$ e dois conjuntos nebulosos

$$\tilde{A}=\{(x,\mu_{\tilde{A}}(x))|\,x\in X\} \quad (10)$$

$$\tilde{B}=\{(y,\mu_{\tilde{B}}(y))|\,y\in Y\} \quad (11)$$

Então $\tilde{R}=\{((x,y),\mu_{\tilde{R}}(x,y))|(x,y)\in X\times Y\}$ é uma relação nebulosa sobre $\tilde{A}$ e $\tilde{B}$ se:

$$\mu_{\tilde{R}}(x,y)\le\mu_{\tilde{A}}(x),\forall(x,y)\in X\times Y \quad (12)$$

e

$$\mu_{\tilde{R}}(x,y)\le\mu_{\tilde{B}}(y),\forall(x,y)\in X\times Y \quad (13)$$

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 dado pelo seguinte conjunto nebuloso:

$$\tilde{R}_1\circ\tilde{R}_2=\{[(x,z),\max_y\{\min\{\mu_{\tilde{R}_1}(x,y),\mu_{\tilde{R}_2}(y,z)\}\}]|x\in X,y\in Y,z\in Z\} \quad (14)$$

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.

A Lógica Nebulosa é uma aplicação da teoria dos conjuntos nebulosos. Dentro do estudo da Lógica Nebulosa e do raciocínio aproximado é bastante utilizado um objeto conhecido 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.

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.

Como exemplo, pode-se citar a variável lingüística com rótulo ***x=velocidade***, com conjunto de termos ***T(velocidade)= {muito_lenta,lenta,rápida,muito_rápida}***, universo de discurso ***U*=[10Km/h,120Km/h]** e um dos valores $\tilde{M}(X)$ como mostrado abaixo.

$$\tilde{M}(lenta)=\{(u,\mu_{lenta}(u))\,|\,u\in[10Km/h,120Km/h]\} \quad (15)$$

A Lógica Nebulosa normalmente é empregada na construção dos chamados Sistemas de Lógica Nebulosa (SLN) (Mendel,

1995), representados pela figura 2. Nesses sistemas, são fornecidas entradas precisas para um módulo ***codificador***, que, por sua vez, fornece parâmetros nebulosos para um máquina 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 ***decodificado***, obtendo-se, dessa forma, a saída precisa do sistema.

Neste artigo, foi preferível usar os termos codificação e decodificação, como tradução da terminologia ***encoding*** e ***decoding***, alternativa para ***fuzzification*** e ***defuzzification***, utilizada, por exemplo, por Pedrycz e Gomide (1998).

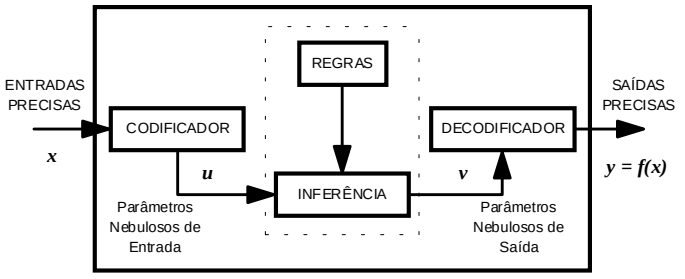

Figura 2 – Sistema de Lógica Nebulosa.

O ***codificador*** mapeia um valor preciso (***crisp***) $x \in X$ para um conjunto nebuloso $\tilde{A}$ em X. Um parâmetro preciso pode ser ***codificado*** seguindo o método ***singleton***, trapezoidal, triangular ou gaussiano. O método mais usado é o ***singleton*** que nada mais é do que criar o conjunto nebuloso $\tilde{A}$ com sua função de pertinência assumindo valor **1** no ponto de X em que o dado parâmetro tem a sua definição e, nos demais pontos, o valor da pertinência seria **0**.

O método ***singleton*** nem sempre é adequado, especialmente quando os dados são corrompidos por ruídos em processos de medidas, por essa razão foram propostos outros três métodos (Mendel, 1995): do triângulo, da curva gaussiana e do trapézio.

A ***decodificação*** pode ser feita, por exemplo, através do valor máximo, da média de valores máximos ou por meio do cálculo do centróide (Mendel, 1995). O método do centróide, baseia-se no cálculo do centro de gravidade conforme a expressão mostrada abaixo.

$$\overline{x} = \frac{\int_S x\mu_{\tilde{A}}(x)dx}{\int_S \mu_{\tilde{A}}(x)dx} \qquad (16)$$

Na expressão (16), $\overline{x}$ é o valor ***decodificado*** de uma dado parâmetro que corresponde ao centro de gravidade (centróide) do conjunto nebuloso com função de pertinência $\mu_{\tilde{A}}(x)$ e suporte S.

Nos sistemas lógicos multivalentes, as proposições podem ter seus valores verdade avaliados não apenas como ***falso*** e ***verdadeiro***, como no caso dos sistemas lógicos bivalentes, mas podem assumir, também, valores lógicos intermediários que podem ser elementos de um conjunto finito ou infinito de valores verdade.

Muitos autores usam a expressão ***lógica nebulosa*** para designar algumas lógicas multivalentes em oposição ao sentido dado por Zadeh (1965) como lógica do raciocínio aproximado. O que se conclui, nesse sentido, é que os valores verdade passam a não ser mais apenas verdadeiro ou falso, mas também a ser uma questão de gradação (Dubois e Prade, 1980). Considerando-se esses conjuntos de valores verdade como o intervalo contínuo [0,1], o valor verdade de uma proposição **P** será, portanto $\upsilon(\mathbf{P}) \in \mathbf{[0,1]}$.

Cada teoria de conjuntos nebulosos definida pelos operadores relativos às operações com conjuntos tem uma lógica multivalente subjacente. Para a teoria de conjuntos nebulosos definida pelos operadores ***max-min-complemento*** em operações mostradas anteriormente, tem-se a lógica subjacente descrita a seguir, onde **P**, **Q** e **R** são proposições e $\upsilon(\mathbf{P})$, $\upsilon(\mathbf{Q})$ e $\upsilon(\mathbf{R})$ são os seus respectivos valores verdades.

O valor verdade da ***negação*** de uma proposição **P**, denotado por **~P**, é dado por:

$$\upsilon(\sim\mathbf{P}) = 1 - \upsilon(\mathbf{P}) \qquad (17)$$

O valor verdade da ***conjunção*** ($\wedge$) e da ***disjunção*** ($\vee$) de duas proposições é dado por:

$$\upsilon(\mathbf{P}\wedge\mathbf{Q}) = min\,[\upsilon(\mathbf{P}), \upsilon(\mathbf{Q})] \qquad (18)$$

$$\upsilon(\mathbf{P}\vee\mathbf{Q}) = max\,[\upsilon(\mathbf{P}), \upsilon(\mathbf{Q})] \qquad (19)$$

Para o conectivo de implicação, existem várias definições da função semântica (Dubois e Prade, 1980; Terano *et alii*, 1987; Lee, 1990). Por exemplo, pela ***regra de implicação booleana*** R_b:

$$\upsilon(\mathbf{P} \xrightarrow{R_b} \mathbf{Q}) = \upsilon(\sim\mathbf{P}\vee\mathbf{Q}) \qquad (20)$$

A definição de equivalência correspondente à expressão anterior seria:

$$\upsilon(\mathbf{P} \xleftrightarrow{R_b} \mathbf{Q}) = \upsilon(\mathbf{P} \xrightarrow{R_b} \mathbf{Q}) \wedge \upsilon(\mathbf{Q} \xrightarrow{R_b} \mathbf{P}) =$$

$$= \upsilon(\sim\mathbf{P}\vee\mathbf{Q}) \wedge \upsilon(\sim\mathbf{Q}\vee\mathbf{P}) \qquad (21)$$

Uma outra definição de implicação é referida como ***regra de implicação aritmética*** R_a:

$$\upsilon(\mathbf{P} \xrightarrow{R_a} \mathbf{Q}) = min\,[1, 1 - \upsilon(\mathbf{P}) + \upsilon(\mathbf{Q})] \qquad (22)$$

$$\upsilon(\mathbf{P} \xleftrightarrow{R_a} \mathbf{Q}) = 1 - |\,\upsilon(\mathbf{P}) - \upsilon(\mathbf{Q})\,| \qquad (23)$$

Essa definição de implicação corresponde à noção de inclusão de conjuntos nebulosos e, juntamente com os operadores ***max-min-complemento***, formam a lógica multivalente subjacente à teoria de conjuntos nebulosos conforme definida por Zadeh (1965) e por Zadeh (1978).

A definição mais amplamente usada nos sistemas especialistas nebulosos é referida como ***regra de mini-operação*** R_m:

$$\upsilon(\mathbf{P} \xrightarrow{R_m} \mathbf{Q}) = min\,[\upsilon(\mathbf{P}), \upsilon(\mathbf{Q})] \qquad (24)$$

A seguinte inequação existe entre os operadores de implicação:

$$\forall\,\mathbf{P}, \mathbf{Q}, \upsilon(\mathbf{P} \xrightarrow{R_a} \mathbf{Q}) \geq \upsilon(\mathbf{P} \xrightarrow{R_b} \mathbf{Q}) \geq \upsilon(\mathbf{P} \xrightarrow{R_m} \mathbf{Q}) \qquad (25)$$

Um sistema de lógica nebulosa típico será composto de várias regras de inferência, manipulando termos de algumas variáveis lingüísticas:

Na lógica nebulosa, utiliza-se uma regra de inferência chamada de ***modus ponens generalizado***, apresentada pela expressão a seguir:

$$\frac{A \supset B}{\frac{A'}{B'}} \qquad (26)$$

A expressão (26) mostra que a pergunta ***A'*** não é exatamente igual ao antecedente ***A*** do axioma ou regra, na verdade, a pergunta é "parecida" com o antecedente, o que leva então a concluir, que a resposta ***B'*** também será "parecida" com o conseqüente ***B*** da implicação.

No caso de se ter várias regras com dois antecedentes, existem dois conjuntos nebulosos de entrada para comparar com cada um dos antecedentes das regras.

A aplicação da inferência ***modus ponens generalizado*** ao caso de várias regras é feita por intermédio de um cálculo que tem início com a determinação do grau de aderência α que as entradas (pergunta) possuem com cada uma das regras. Essa aderência atenua ou não (dependendo do seu valor) a influência do conseqüente de cada regra com o resultado que, por sua vez, é uma combinação de todos os conseqüentes de todas as regras. Tal cálculo, apresentado detalhadamente em pseudo-código, segue os seguintes passos para um exemplo com ***n*** regras de dois antecedentes $\tilde{A}_i$ e $\tilde{B}_i$ e um conseqüente $\tilde{C}_i$, onde ***i*** assume valores de ***1*** até ***n***:

(1) Sejam $\tilde{A}$ e $\tilde{B}$ dois conjuntos nebulosos representando um par de valores de entrada. Sejam também $\mu_{\tilde{A}}$ e $\mu_{\tilde{B}}$ as funções de pertinência de $\tilde{A}$ e $\tilde{B}$ respectivamente.
*(2) Para cada regra **Ri** Faça (**i = 1, 2, ..., n**):*
(2.1) Achar a função mínima entre a função de pertinência do antecedente $\tilde{A}_i$ e $\mu_{\tilde{A}}$, nomeando-a de $\mathbf{f_{mínA}}$;
(2.2) Achar a função mínima entre a função de pertinência do antecedente $\tilde{B}_i$ e $\mu_{\tilde{B}}$, nomeando-a de $\mathbf{f_{mínB}}$;
(2.3) $\boldsymbol{\alpha_i}$ = mínimo (supremo ($\mathbf{f_{mínA}}$), supremo ($\mathbf{f_{mínB}}$));
*(2.4) Achar a função mínima entre a função constante **f** = $\boldsymbol{\alpha_i}$ e o conseqüente de **Ri**, nomeando-a de $\mathbf{f_{mínCi}}$;*
(3) Fim do Para .... Faça
*(4) Achar a função máxima entre todas as funções $\mathbf{f_{mínCi}}$ para **i** variando de **1** à **n**.*

No exemplo considerado, para expor um método de inferência dos sistemas de lógica nebulosa, as duas entradas $\tilde{A}_i$ e $\tilde{B}_i$ seriam resultados de uma ***codificação*** de valores precisos de determinada entrada. A resposta para essas duas entradas seria um conjunto nebuloso com universo de discurso da variável lingüística do conseqüente $\tilde{C}_i$ e com função de pertinência obtida no passo (4). Procedendo-se então com a ***decodificação*** de tal resposta, obter-se-ia finalmente um valor preciso da resposta desejada.

O presente trabalho foi desenvolvido, usando dois métodos de inferência: 1) a composição de relações nebulosas (Zimmermann, 1985); e 2) ***modus ponens generalizado***. Esse último método é obtido a partir do primeiro (inferência composicional). O operador de implicação escolhido foi o $\boldsymbol{R_a}$ (regra de implicação aritmética) que pode ser visto pela expressão (22). Tal operador foi escolhido por ser o mais abrangente (expressão 25).

O exemplo acima mostrou, em pseudo-código, o cálculo da inferência com base na regra ***modus ponens generalizado***. Esse pseudo-código foi deduzido com a aplicação do operador $\boldsymbol{R_a}$ ao caso de várias regras de um sistema de lógica nebulosa. A dedução completa pode ser vista em (Bernardo Filho, 1999).

2.3 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 considerado. Tais técnicas de busca fazem parte de uma categoria de métodos usados pela Inteligência Artificial para resolver problemas que não possuem uma solução determinística.

No caso de especificações de protocolos de comunicação feitas com um modelo de estados e transições, uma dada técnica de busca é aplicada dentro do espaço de estados globais de um protocolo, cujos nós são obtidos por meio dos disparos das transições do modelo utilizado.

Os métodos que determinam a busca dos estados objetivos em problemas de Inteligência Artificial são chamados de ***estratégias de controle***. Dois tipos básicos de estratégias de controle podem ser mencionados: o regime irrevogável e o regime de controle por tentativas. No regime irrevogável, uma transição é disparada sem previsão para reconsideração futura, ou seja, a computação de um estado produzido por essa transição não é armazenada. No regime de controle por tentativa, leva-se em conta a possibilidade de retornar a um determinado ponto da computação e executar outra transição, caso o caminho analisado não se mostre promissor.

Os métodos heurísticos usualmente empregam funções de avaliação para determinar o quanto um nó é promissor. A escolha adequada da função de avaliação é crítica na determinação dos resultados da procura. Caso essa função falhe em reconhecer a potencialidade de alguns nós, serão obtidos caminhos de custo não mínimo e, por outro lado, o uso de uma função que superestime a potencialidade de vários nós resulta na expansão desnecessária dos caminhos a eles associados.

Neste trabalho, o algoritmo **A*** (Rich e Knight, 1991) 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, pois 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 técnica de verificação usa, na realidade, uma modificação do algoritmo **A*** o qual incorpora a execução de dois sistemas de lógica nebulosa, sendo aqui denominado de algoritmo **Ã***.

No algoritmo **A***, os nós recentemente criados são armazenados na fila de nós abertos. A fila de *ABERTOS*, no algoritmo **A*** original, mantém os nós ordenados de maneira crescente, segundo os valores de uma função de tal forma que o primeiro nó da fila é aquele que possui o menor custo para o objetivo. Na técnica usada pelo **Verest+**, a fila de *ABERTOS* é ordenada por intermédio do processamento de regras de um Sistema de Lógica Nebulosa (SLN). O algoritmo **A*** é apresentado em pseudo-código no apêndice A (seção 8).

No algoritmo **A***, o passo (ou subalgoritmo) da linha 82 ***Colocar_ordenado_em_aberto (Suc)*** tem por objetivo armazenar

os nós recentemente criados na fila de nós abertos. A fila de *ABERTOS*, 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.

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.

3 A TÉCNICA DE VERIFICAÇÃO

A técnica de verificação usada pelo **Verest+** (Bernardo Filho *et alii*, 1998a; Bernardo Filho *et alii*, 1998b) 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.

O emprego da Lógica Nebulosa (Zimmermann, 1985; Mendel, 1995; Zadeh, 1965) 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 chamado de Sistema sob Estudo (**SE**), o qual deverá ser especificado pela linguagem ESTELLE.

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 idéia de se comparar os dois sistemas (**SE** e **SP**) está calcada na observação das normas de alguns protocolos de diversos modelos de referências em que se pode constatar a existência de funções semelhantes em vários desses protocolos, como, por exemplo, o fato de aparecer as funções de gerência de conexão, controle fluxo e controle de erros entre outras.

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 do 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:

- $\mathbf{E} = \{e_1,...,e_{n1}\}$ é o conjunto dos estados parciais de **SE**;
- $\mathbf{X} = \{x_1,...,x_{n2}\}$ é o conjunto dos estados parciais de **SP**.

A partir de então, define-se a relação nebulosa $\tilde{R}_{EX}(e,x)$ (Zimmermann, 1985) 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 (**Ã***) 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, para se obter um estado parcial **x** em **SP**, que corresponda a um estado parcial **e** em **SE**, deve ser feita a seguinte composição de relações nebulosas:

$$\underline{\tilde{O}}(x) = \tilde{O}(e) \circ \tilde{R}_{EX}(e,x) \qquad (27)$$

onde

- **$\tilde{O}$(e)** é um estado parcial $\mathbf{e_i}$ do **SE** ***codificado*** pelo método ***singleton*** (Mendel, 1995);
- **$\underline{\tilde{O}}$(x)** é o estado parcial nebuloso em **SP**, equivalente a **$\tilde{O}$(e)** em **SE**;
- $\mathbf{x_k} \in \mathbf{X}$ é obtido da ***decodificação*** pelo método do máximo de **$\underline{\tilde{O}}$(x)**.

A equação (27) 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 (Zimmermann, 1985) ***semelhança***, cujo universo de discurso ***U*** é composto pelos valores da função de pertinência da relação $\tilde{R}_{EX}(e,x)$. Na figura 3, pode-se observar a definição dessa variável lingüística.

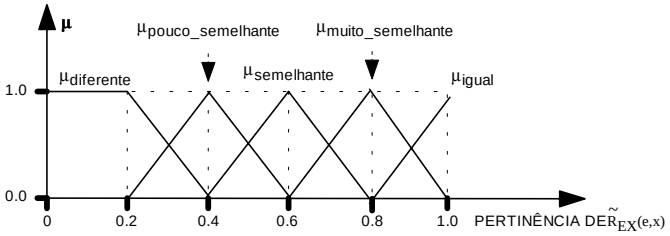

Figura 3 – Funções de pertinência dos termos da variável lingüística *semelhança***.**

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 (27) para cada um dos estados parciais $\mathbf{e_1}$ e $\mathbf{e_2}$, chegou-se à conclusão de que os seus equivalentes, deveriam ser $\mathbf{x_3}$ e $\mathbf{x_5}$ no **SP**. No entanto, o par $(\mathbf{x_3},\mathbf{x_5})$ 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 ***decodificados*** $\mathbf{x_3}$ ou $\mathbf{x_5}$, e formar 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 globais em **SP** que seja um estado global e ainda seja o mais próximo de $(\mathbf{e_1},\mathbf{e_2})$.

O subalgoritmo da linha 82 ***Colocar_ordenado_em_aberto (Suc)*** do algoritmo **Ã*** aloca os nós na fila de *ABERTOS* a partir de um SLN que manipula termos de uma outra variável lingüística, chamada ***posição***, cuja definição aparece na figura 4. Nessa figura, observa-se o universo de discurso da variável caracterizado pelo percentual de nós na fila de *ABERTOS*, pois, conforme **Ã*** evolui, a quantidade de estados globais obtidos aumenta, sendo então necessário, inicialmente, ***decodificar*** um percentual para em seguida calcular a posição

atual.

Além das variáveis lingüísticas ***semelhança*** e ***posição***, é definida também a variável ***custo*** (veja figura 5), que atribui termos nebulosos ao custo do nó do **SP**, equivalente a 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**.

O custo do nó em **SP,** equivalente ao nó sucessor, para o objetivo do **SE**, deve ser ***codificado*** segundo o método ***singleton*** (Mendel, 1995). 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 *ABERTOS* (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**.

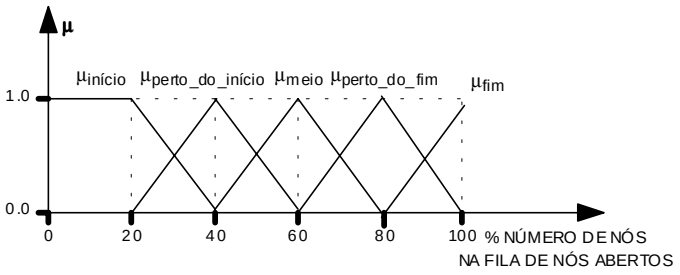

Figura 4 – Funções de pertinência dos termos da variável lingüística *posição***.**

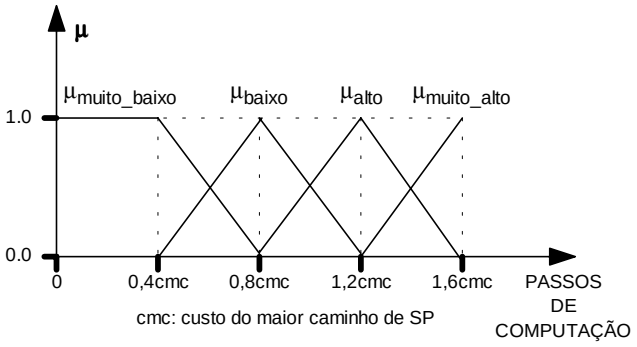

Figura 5 – Funções de pertinência dos termos da variável lingüística *custo***.**

Algumas regras do SLN A 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, $\mathbf{x_k}$ e $\mathbf{x_l}$, compondo o estado global do **SP** equivalente ao do **SE**.

- *SE $\{e_i,x_k\}$ É igual E $\{e_j,x_l\}$ É igual ENTÃO $\{(e_i,e_j),(x_k,x_l)\}$ É igual*
- *SE $\{e_i,x_k\}$ É igual E $\{e_j,x_l\}$ É muito_semelhante ENTÃO $\{(e_i,e_j),(x_k,x_l)\}$ É muito_semelhante*
- *SE $\{e_i,x_k\}$ É muito_semelhante E $\{e_j,x_l\}$ É igual ENTÃO $\{(e_i,e_j),(x_k,x_l)\}$ É muito_semelhante*
- *SE $\{e_i,x_k\}$ É igual E $\{e_j,x_l\}$ É pouco_semelhante ENTÃO $\{(e_i,e_j),(x_k,x_l)\}$ É semelhante*
- *SE $\{e_i,x_k\}$ É igual E $\{e_j,x_l\}$ É diferente ENTÃO $\{(e_i,e_j),(x_k,x_l)\}$ É semelhante*
- *SE $\{e_i,x_k\}$ É pouco_semelhante E $\{e_j,x_l\}$ É igual ENTÃO $\{(e_i,e_j),(x_k,x_l)\}$ É semelhante*
- *SE $\{e_i,x_k\}$ É pouco_semelhante E $\{e_j,x_l\}$ É diferente ENTÃO $\{(e_i,e_j),(x_k,x_l)\}$ É diferente*
- *SE $\{e_i,x_k\}$ É diferente E $\{e_j,x_l\}$ É diferente ENTÃO $\{(e_i,e_j),(x_k,x_l)\}$ É diferente*

Algumas regras de inferência do SLN B ordenador são apresentadas logo a seguir, onde $\boldsymbol{\eta_s}$ é o estado sucessor de *MELHOR_NÓ* do **SE**, equivalente ao nó $\boldsymbol{\sigma_s}$ do **SP**; $\tilde{C}o(\sigma_s)$ é o custo nebuloso de $\boldsymbol{\sigma_s}$ ao nó $\boldsymbol{\sigma_o}$ (objetivo no **SP**); e, $\tilde{P}(\eta_s)$ é a posição nebulosa de $\boldsymbol{\eta_s}$ na fila de *ABERTOS*.

- *SE ($\{\boldsymbol{\eta_s},\boldsymbol{\sigma_s}\}$É igual) E ($\tilde{C}o(\sigma_s)$ É muito_baixo) ENTÃO $\tilde{P}(\eta_s)$ É início*
- *SE ($\{\boldsymbol{\eta_s},\boldsymbol{\sigma_s}\}$É igual) E ($\tilde{C}o(\sigma_s)$ É baixo) ENTÃO $\tilde{P}(\eta_s)$ É perto_do_início*
- *SE ($\{\boldsymbol{\eta_s},\boldsymbol{\sigma_s}\}$É igual) E ($\tilde{C}o(\sigma_s)$ É alto) ENTÃO $\tilde{P}(\eta_s)$ É meio*
- *SE ($\{\boldsymbol{\eta_s},\boldsymbol{\sigma_s}\}$É igual) E ($\tilde{C}o(\sigma_s)$ É muito_alto) ENTÃO $\tilde{P}(\eta_s)$ É fim*
- *SE ($\{\boldsymbol{\eta_s},\boldsymbol{\sigma_s}\}$É muito_semelhante) E ($\tilde{C}o(\sigma_s)$ É muito_alto) ENTÃO $\tilde{P}(\eta_s)$ É perto_do_fim*
- *SE $\{\boldsymbol{\eta_s},\boldsymbol{\sigma_s}\}$ É diferente ENTÃO $\tilde{P}(\eta_s)$ É fim*

A técnica de verificação, portanto, pode ser vista como uma comparação entre um conhecimento existente (sistema padrão) e um conhecimento a ser adquirido (sistema sob estudo). O usuário da técnica fornece a semelhança parcial entre os dois sistemas, enquanto a inferência nebulosa acha a semelhança global. Na medida em que é feita a busca em um espaço de estados, para alcançar um objetivo, a semelhança global é maximizada.

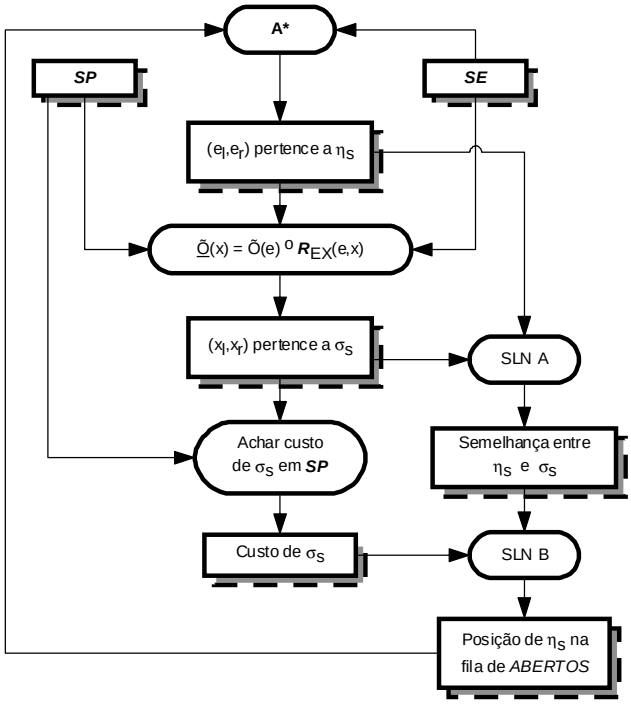

Figura 6 – A técnica de verificação para protocolos de comunicação.

A figura 6 resume toda a técnica de verificação, ou seja, a cada passo o algoritmo **A*** gera um novo estado global $\boldsymbol{\eta_s}$ constituído por, entre outros objetos, um par de estados parciais $(\mathbf{e_l},\mathbf{e_r})$ das entidades local e remota, respectivamente, envolvidas na utilização da heurística. Ao aplicar a composição de relações nebulosas dada pela expressão (27), obtém-se o par de estados parciais $(\mathbf{x_l},\mathbf{x_r}) \in \boldsymbol{\sigma_s}$ no sistema padrão **SP** equivalente ao par $(\mathbf{e_l},\mathbf{e_r})$.

O Sistema de Lógica Nebulosa A (SLN A) mede a semelhança entre os estados globais $\boldsymbol{\eta_s}$ e $\boldsymbol{\sigma_s}$. A semelhança medida, assim

como o custo de σ_s ao objetivo no **SP** são fornecidos como entradas para o Sistema de Lógica Nebulosa B (SLN B) que acha a posição na fila de *ABERTOS* do algoritmo **A***. O processo continua com o algoritmo **A*** gerando mais um estado global η_s até que o nó objetivo seja alcançado ou até que a fila de *ABERTOS* esteja vazia (árvore de alcançabilidade esgotada).

O algoritmo **Ã***, implementado no **Verest+,** não faz a computação do custo **g** anteriormente obtido até um dado nó em análise. Isso ocorre porque o algoritmo **Ã*** obtém a posição nebulosa na fila de *ABERTOS* diretamente a partir da execução do SLN B, ou seja, a fila de *ABERTOS* não é ordenada pela função **f'** = **g** + **h'** e sim pelo resultado da inferência do SLN B. O pseudo-código apresentado na seção anterior retrata o algoritmo **A*** original e, logo, aparece a manipulação do custo **g**, mas no algoritmo **Ã***, tal manipulação não está presente.

4 IMPLEMENTAÇÃO DO VEREST+

4.1 Arquitetura do Verest+

A figura 7, a seguir, apresenta a arquitetura do **Verest+**, implementado com base na técnica descrita na seção anterior. O **Verest+** foi desenvolvido com a linguagem MODULA-2, possuindo ao todo 120 módulos com aproximadamente 30 000 linhas de código. O bloco ***A*** representa o seu usuário que tem como função elaborar os arquivos de entrada para o verificador (blocos ***B***, ***C***, ***D*** e ***E***).

O arquivo do bloco ***B*** traz a definição dos dados para o processamento da inferência nebulosa através da execução do SLN (Sistema de Lógica Nebulosa) A e B. O bloco ***C*** representa o arquivo da forma intermediária estática gerado pelo compilador ESTELLE (Oliveira Jr., 1991).

O arquivo caracterizado pelo bloco ***D*** contém a definição dos dados referentes à propriedade a ser testada pelo verificador, ou seja, a fórmula de lógica temporal que descreve a propriedade, os operandos dessa fórmula e o estado global objetivo.

O último arquivo de entrada, representado pelo bloco ***E***, contém o grafo do sistema padrão a ser utilizado na comparação com o sistema sob estudo. Esse arquivo possui o registro do custo de cada nó do **SP** a todos os seus outros nós.

Após a leitura do arquivo com os dados usados pelos sistemas de lógica nebulosa (bloco ***B***), as suas informações são armazenadas nos módulos integrantes dos blocos ***F*** (dados da lógica nebulosa) e ***J*** (dados gerais da verificação). O bloco ***G*** (dados estáticos de ESTELLE) é composto pelos módulos que armazenam as informações lidas do arquivo da forma intermediária estática, enquanto o bloco ***H*** (dados dinâmicos de ESTELLE) registra durante a verificação, as informações variáveis do sistema sob estudo.

O arquivo do bloco ***D***, tem as suas informações armazenadas nos blocos ***J*** e ***I*** (dados da lógica temporal). O bloco ***L*** representa o máquina de inferência dos Sistemas de Lógica Nebulosa A e B que é chamado pelo executor do algoritmo **Ã***, caracterizado pelo bloco ***N***. Para poder processar o algoritmo **Ã***, o bloco ***N*** precisa interagir também com os blocos ***M*** (forma intermediária dinâmica) e ***O*** (executor das primitivas de ESTELLE). A forma intermediária dinâmica é responsável pela execução das ações que ocorrem após o disparo das transições da especificação do **SE**, bem como da avaliação da função da condição ***booleana*** presente nessas mesmas transições. O bloco ***O*** é composto pela implementação das primitivas de ESTELLE.

Os blocos ***P***, ***Q***, ***R*** e ***S*** simbolizam os arquivos de saída, respectivamente, arquivo de descrição dos resultados, arquivo que registra a seqüência de disparo de transições do caminho analisado, arquivo que possui o grafo parcial do sistema sob estudo, obtido na análise, e, arquivo que possui uma descrição textual dos estados globais do **SE** expandidos durante a análise.

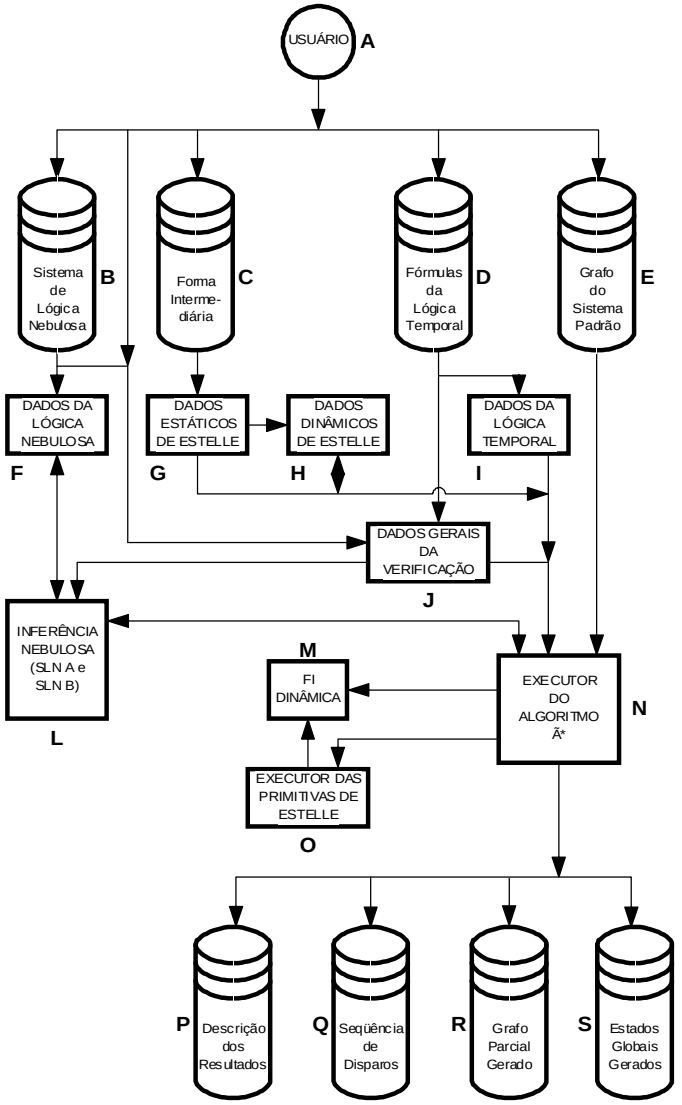

Figura 7 – Estrutura modular do verificador Verest+.

4.2 Sistema Padrão

O Sistema Padrão deve ser fornecido pelo usuário na forma de um arquivo binário (bloco ***E***) que contém toda a sua árvore de alcançabilidade. A árvore de alcançabilidade do **SP** pode ser obtida por qualquer ferramenta de verificação que faça uma busca exaustiva como, por exemplo, o verificador comentado em (Bernardo Filho *et alii*, 1992). O arquivo do Grafo do Sistema Padão (***GSP***) pode ser criado com o auxílio do programa **CADSP**.

O arquivo ***GSP*** (bloco ***E***) possui formato binário e armazena valores do tipo inteiro (dois ***bytes***) que denotam a distância, em números de disparos de transições, entre dois estados globais do **SP**. Os estados globais do **SP** considerados pela técnica de verificação levam em conta apenas a composição de estados parciais de duas instâncias de módulo da especificação do **SP**. Portanto, o arquivo ***GSP*** armazena as distâncias entre dois pares de estados parciais, pois cada par de estados parciais corresponde a um estado global de **SP**.

No sentido de esclarecer melhor a estrutura do arquivo ***GSP***, seja o seguinte exemplo em que o **SP** é constituído por uma instância de módulo (local) cujo corpo possui três estados parciais (l_1, l_2 e l_3) e por uma outra instância de módulo (remota) cujo corpo possui dois estados parciais (r_1 e r_2). Existem, nesse caso, 3×2=6 pares possíveis de estados parciais e, logo, existem seis estados globais possíveis de ocorrer em **SP**. O arquivo ***GSP*** armazena a distância, em número de disparos de transições, que cada um desses pares leva para atingir todos os outros.

Caso a especificação de **SP** não possua um determinado par de estados parciais $\boldsymbol{\sigma_x}$ como estado global, então todas as distâncias dos outros pares de estados parciais, que são estados globais, a esse par $\boldsymbol{\sigma_x}$ devem ser registradas, no arquivo ***GSP***, com um valor inteiro negativo (-1, por exemplo), de forma a deixar claro que aqueles caminhos aos pares de estados parciais que não são estados globais não existem.

As distâncias dos pares de estados parciais, que são estados globais, a eles próprios são registradas no arquivo ***GSP*** com o valor zero. Uma maneira de se verificar, por exemplo, se um dado par de estados parciais $\boldsymbol{\sigma_x}$ caracteriza um estado global em **SP**, é a distância $\boldsymbol{d}$ ($\boldsymbol{\sigma_x}$, $\boldsymbol{\sigma_x}$) dele para ele mesmo. Caso seja zero, $\boldsymbol{\sigma_x}$ é um estado global; caso $\boldsymbol{d}$ ($\boldsymbol{\sigma_x}$, $\boldsymbol{\sigma_x}$) seja menor que zero, então $\boldsymbol{\sigma_x}$ não é um estado global.

Pelo que foi explicado anteriormente, existem informações desnecessárias no arquivo ***GSP***, ou seja, distâncias a estados globais que não fazem parte da execução do **SP**. Entretanto, foi decidido manter tais informações para que a busca das distâncias entre os estados globais, assim como a identificação de um dado par de estados parciais como estado global, ficasse mais rápida.

Caso fossem armazenados apenas os pares de estados parciais que são estados globais, na ocasião em que tivesse sido obtido um possível estado global $\boldsymbol{\sigma}$ do **SP** equivalente ao estado global atual $\boldsymbol{\eta}$ do **SE** (durante a execução do algoritmo **Ã***, para verificar se $\boldsymbol{\sigma}$ é de fato um estado global de **SP)**, seria necessário processar várias comparações de $\boldsymbol{\sigma}$ com os estados do arquivo ***GSP***, a fim de identificar se o mesmo seria um estado global.

Ao se colocar todas as distâncias a todos os pares de estados parciais de **SP** no arquivo ***GSP***, mesmo daqueles que não são estados globais de **SP**, torna-se mais fácil, além de mais rápido, processar o acesso a essas distâncias. Alocando-se no arquivo do ***GSP*** as distâncias conforme os identificadores dos estados parciais, para recuperar tais informações, basta usar uma fórmula para se obter a posição de leitura de uma certa distância no arquivo.

Seguindo o exemplo hipotético em que o **SP** é constituído por uma instância de módulo (local) cujo corpo possui três estados parciais (l_1, l_2 e l_3) e por uma outra instância de módulo (remota) cujo corpo possui dois estados parciais (r_1 e r_2), o conteúdo do arquivo ***GSP*** ficaria da forma como é mostrada na tabela 2.

Para entender a tabela 2, é necessário lembrar que existem seis estados globais possíveis no exemplo considerado (3 locais × 2 remotos) e, ao se armazenar a distância de cada um desses estados a todos os outros, tem-se 6×6=36 distâncias como conteúdo do arquivo ***GSP***. Os estados parciais locais são numerados de 1 a 3, enquanto os estados remotos são numerados de 1 a 2. Assim, o primeiro par seria (1,1), o segundo (1,2) e assim por diante até (3,2). Cada um desses pares tem uma distância definida para todos os outros conforme mostrado, esquematicamente, na tabela 2. Em primeiro lugar, são registradas as distâncias de cada par ao par (1,1), passando-se depois ao registro das distâncias de cada par ao par (1,2) e assim por diante.

Tabela 2 – Conteúdo de um exemplo do arquivo *GSP*.

POSIÇÃO	PAR ORIGEM DO CAMINHO	PAR DESTINO DO CAMINHO	DISTÂNCIA
1	(1,1)	(1,1)	D1
2	(1,2)	(1,1)	D2
:	:	:	:
7	(1,1)	(1,2)	D7
8	(1,2)	(1,2)	D8
:	:	:	:
17	(3,1)	(2,1)	D17
18	(3,2)	(2,1)	D18
:	:	:	:
35	(3,1)	(3,2)	D35
36	(3,2)	(3,2)	D36

A tabela 2 destaca alguns dos registros no arquivo ***GSP*** do exemplo hipotético. Nela pode-se ver as duas primeiras distâncias (D1 e D2), as duas últimas (D35 e D36) e quatro distâncias intermediárias (D7, D8, D17 e D18). Para se obter a posição dessas distâncias a partir dos identificadores numéricos dos estados parciais, devem-se empregam-se as seguintes fórmulas:

$$Posição\ [(x_r,x_s);(x_t,x_u)] =$$

$$= (Pos1\ (x_t,x_u) - 1) \times TotPares + Pos1\ (x_r,x_s) \qquad (28)$$

$$Pos1\ (x_i,x_j) = j + (i - 1) \times TotRemotos \qquad (29)$$

Na fórmula (28), $\boldsymbol{(x_r,x_s)}$ e $\boldsymbol{(x_t,x_u)}$ são dois pares de estados parciais quaisquer do **SP**, enquanto ***TotPares*** é o total de pares de estados parciais, ou seja, o produto do maior identificador (numérico) dos estados parciais locais pelo maior identificador (numérico) dos estados parciais remotos. Na fórmula (29), $\boldsymbol{(x_i,x_j)}$ é um par de estados parciais qualquer de **SP** e ***TotRemotos*** é o total de estados parciais remotos, o que corresponde também ao valor do maior identificador dos estados parciais remotos.

A fim de demonstrar o emprego das fórmulas (28) e (29), a posição da distância entre os pares de estados parciais (3,2) e (2,1) é calculada da seguinte forma:

$$Pos1\ (3,2) = 2 + (3 - 1) \times 2 = 6 \quad [29 \text{ aplicada para } (3,2)]$$

$$Pos1\ (2,1) = 1 + (2 - 1) \times 2 = 3 \quad [29 \text{ aplicada para } (2,1)]$$

$$Posição\ [(3,2);(2,1)] = (Pos1\ (2,1) - 1) \times 6 +$$

$$Pos1\ (3,2) = (3 - 1) \times 6 + 6 = 18$$

[28 aplicada a (3,2) e (2,1)]

Como se pode ver, a posição obtida é 18, o que está de acordo com a tabela 2. Entretanto, para se obter a posição real da distância D18 entre os pares (3,2) e (2,1) no arquivo ***GSP*** deve-se levar em conta, ainda, o tamanho, em ***bytes***, do valor da distância e também lembrar que o primeiro ***byte*** do arquivo ***GSP*** está na posição zero. Portanto, a fórmula para a posição real da distância entre dois pares de estados parciais $\boldsymbol{p_i}$ e $\boldsymbol{p_j}$, no arquivo ***GSP***, é dada por:

$$PosiçãoReal\ [p_i;p_j] =$$

$$= (Posição\ [p_i;p_j] - 1) \times Tamanho\ (inteiro) \qquad (30)$$

Na fórmula (30), o operando ***Posição*** é calculado pela equação (28) para os pares p_i e p_j, enquanto o operando ***Tamanho (inteiro)*** é a quantidade de ***bytes*** que o tipo inteiro da linguagem MODULA-2 ocupa na memória.

O programa **CADSP** (cadastrar o sistema padrão) é uma pequena ajuda para criar o arquivo do grafo do sistema padrão (***GSP***), que consiste em uma das entradas para o **Verest+**. O arquivo ***GSP*** armazena a distância (custo) de cada par de estados parciais (estado global) a todos os outros pares (estados globais) do sistema padrão (**SP**). Portanto, o programa **CADSP** precisa saber quais são os estados parciais componentes do **SP** em questão, além das referidas distâncias. Desde que o usuário forneça tais informações ao programa **CADSP**, ele irá gerar o arquivo ***GSP*** no formato exigido pelo **Verest+**.

5 RESULTADOS

5.1 O Sistema Padrão

O sistema padrão (**SP**) utilizado na comparação com os três protocolos verificados que serviram de testes para o **Verest+** é um protocolo simples, envolvendo duas entidades comunicantes. Esse protocolo faz a abertura de uma conexão, transmite dados e, então, encerra a conexão. Na figura 8, pode ser observada a rede de predicado-ação que modela a entidade iniciadora e a entidade respondedora 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**.

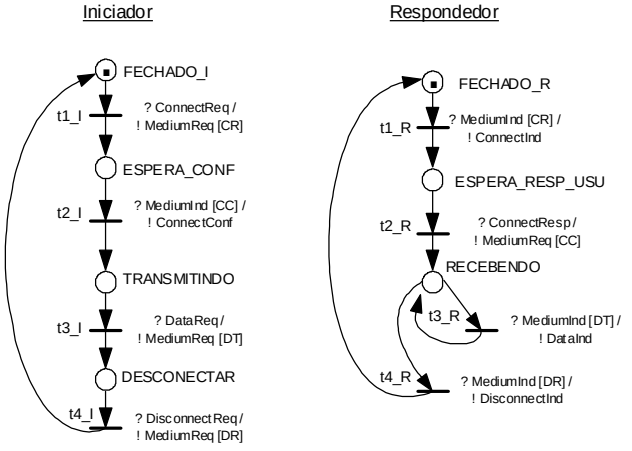

Figura 8 – Modelo dos comportamentos Iniciador e Respondedor do SP.

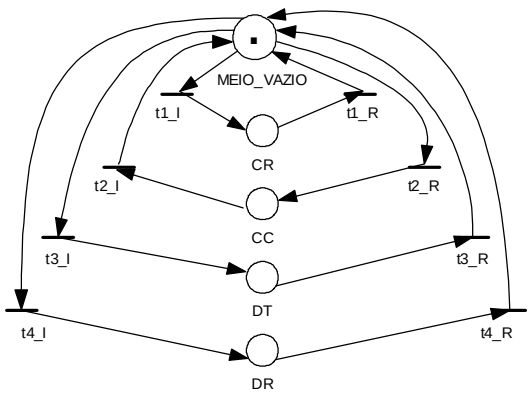

Figura 9 – Modelo do meio de comunicação (*Medium*) entre as duas entidades pares do SP.

A tabela 3 expõe o grafo de execução do sistema padrão feito com o programa ARP 2.3 (Farines, 1989), onde se pode observar, que o estado **M4** é obtido após **4** disparos de transição. Em alguns dos testes realizados com os protocolos escolhidos, o estado **M4** irá caracterizar o objetivo em **SP**. Em outras palavras, σ_o = (*TRANSMITINDO, RECEBENDO*).

No grafo de **SP** (tabela 3), 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 5 (função de pertinência da variável lingüística **custo** — veja seção 3), 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**.

Tabela 3 – 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}

O arquivo ***GSP*** (veja subseção 4.2) foi criado a partir da tabela 3 e com a utilização do programa **CADSP**. Tal arquivo ***GSP*** criado foi o empregado em todos os testes, já que o sistema padrão comparado com os três protocolos foi o mesmo. Foi usado um mesmo sistema padrão, pois os três protocolos considerados possuem, pelo menos, as três fases do **SP** descrito, ou seja: 1) abertura de conexão, 2) transmissão de dados e 3) desconexão.

O sistema padrão deve ser escolhido o mais semelhante possível com o sistema sob estudo, pois só assim será obtido a máxima eficácia da heurística de busca que tem a função de acelerar a procura dos nós objetivos que prova a existência ou ausência de uma dada propriedade do sistema sob estudo. Na medida em que os usuários do **Verest+** forem criando (cadastrando), com o programa **CADSP**, novas especificações de **SP**, os novos usuários passarão a contar com um leque de opções maior de tal forma a facilitar a escolha de um **SP** mais "parecido" com o **SE** em questão.

Caso o sistema padrão escolhido seja muito diferente do sistema sob estudo, os estados globais η_s e σ_s (veja figura 6 na seção 3) seriam diferentes e, logo, a última regra mostrada no SLN B (veja seção 3) teria a maior influência na resposta da posição de η_s na fila de estados globais *ABERTOS* do algoritmo **Ã***, então um novo nó η_s expandido sempre seria colocado no fim da fila, permitindo que os nós expandidos anteriormente fossem analisados primeiro. Se o novo nó η_s estivesse apontando um caminho para o objetivo, este não seria percorrido imediatamente, pois todos os nós expandidos anteriormente a η_s seriam analisados primeiro, mas quando chegasse o momento da análise de η_s, o caminho para o objetivo seria analisado de qualquer forma.

Concluindo, se o **SP** for diferente do **SE** (ausência de informação ou heurística), o **Verest+** continuará encontrando o objetivo procurado, caso o mesmo exista, apenas levará mais tempo. A função da heurística é introduzir conhecimento sobre o **SE** por meio de um **SP** semelhante para que as buscas pelos objetivos sejam mais rápidas.

5.2 Protocolo HMS

O protocolo HMS (***Hierarchical Management System***), usado aqui como teste, é formado pela especificação de um dos serviços de um protocolo desenvolvido para gerência de grandes redes de telecomunicações. Tal protocolo foi proposto por Fernandez e Pedroza (1997), tendo sido baseado em um modelo hierárquico de gerência. A especificação ESTELLE do HMS, que foi testada pelo **Verest+**, implementa a primitiva **PolicyAChange**, responsável pela alteração das diretrizes de um agente ou gerente filho.

A figura 10 mostra a arquitetura da especificação ESTELLE do HMS e a figura 11 apresenta um modelo em rede de Petri que corresponde a essa especificação ESTELLE que pode ser vista completa em Bernardo Filho (1999).

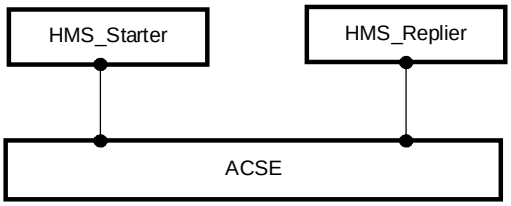

Figura 10 – Arquitetura da especificação ESTELLE do protocolo HMS.

A especificação do HMS possui três instâncias de módulo: ***HMS_Starter*** (iniciador do serviço de alteração de diretrizes), ***HMS_Replier*** (respondedor do serviço de alteração de diretrizes) e ***ACSE*** (meio de comunicação da camada de aplicação).

O módulo ***HMS_Starter*** possui os estados *IDLE*, *WaitCONNECTION* e *WaitTABLEPOLICY*. Ele envia a mensagem de alteração de diretrizes ao respondedor, imediatamente antes de entrar no estado *WaitTABLEPOLICY* a partir do qual passa a aguardar a confirmação do envio.

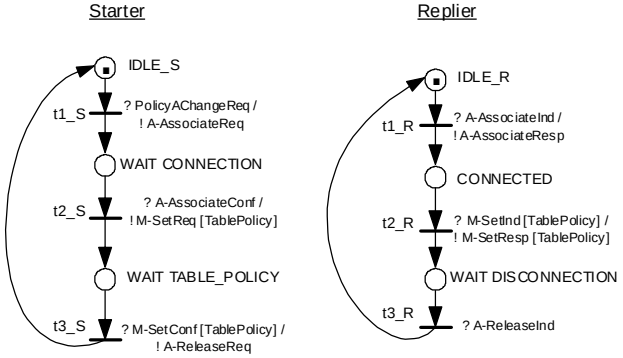

Figura 11 – Modelo da primitiva PolicyAChange.

O módulo ***HMS_Replier*** possui os estados *IDLE*, *CONNECTED* e *WaitDISCONNECT*. Ao entrar no estado *WaitDISCONNECT*, o módulo ***HMS_Replier*** já terá recebido a mensagem de alteração de diretrizes do iniciador. O teste realizado com o protocolo HMS procurou um estado global objetivo η_o que comprovasse a execução do serviço, ou seja, a entrega da mensagem de mudança de diretrizes. Portanto, tal objetivo foi definido pelos módulos ***HMS_Starter*** e ***HMS_Replier*** nos estados parciais, respectivamente, *WaitTABLEPOLICY* e *WaitDISCONNECT*.

A verificação do objetivo η_o do protocolo HMS é apresentada no arquivo de descrição dos resultados. No campo conclusão desse arquivo (mostrado a seguir) pode ser observado que o objetivo foi alcançado em 9 segundos e constatou-se que a fórmula de lógica temporal testada ***S1 u S2*** (veja tabela 1) é verdadeira. A fórmula ***S1 u S2*** foi escolhida, pois desejava-se testar a ocorrência do objetivo η_o, então bastava averiguar se tal objetivo iria ocorrer em algum momento, fazendo ***S1*** verdadeiro e ***S2*** = η_o.

Trecho do arquivo de descrição dos resultados do HMS

```
==========================================
Razao do final: Objetivo alcancado
==========================================
Conclusao:

Total de estados globais obtidos: 8
Tempo de verificacao: 0 horas, 0 minutos e 9 segundos
Resultado da formula -> FORMULA VERDADEIRA
Razao -> S1 tem ocorrido e S2 ocorreu
```

5.3 Protocolo Abracadabra

O Abracadabra (Courtiat, 1987) foi padronizado em conjunto pela ISO e pela ITU (***International Telecommunication Union***) para ser usado apenas em testes. Trata-se de um protocolo orientado a conexão, apresentando características semelhantes às do protocolo do ***bit alternado***, pois suas ***PDUs*** (***protocol data units***) são transmitidas com números de seqüência alternados entre **0** (zero) e **1** (um).

A especificação do protocolo Abracadabra testada possui cinco instâncias de módulo: ***Usuário A***, ***Usuário B***, ***Abracadabra A***, ***Abracadabra B*** e ***Meio DG***. Com relação aos estados das entidades do Abracadabra (***A*** e ***B***), tem-se:

- *CLOSED* — estado inicial em que o protocolo encontra-se com a conexão fechada;
- *WFCC* — após um envio de pedido de abertura de conexão à entidade, o protocolo espera uma ***PDU*** de confirmação da abertura dessa conexão (***PDU CC – connect confirm***);
- *OPEN* — estado em que o protocolo encontra-se com a conexão aberta;
- *WFAK* — nesse estado, o protocolo espera o reconhecimento do envio de uma ***PDU*** de dado, durante a fase de transmissão de dados;
- *CLOSING* — estado onde o protocolo encontra-se encerrando a conexão;
- *ERROR* — estado de erro do protocolo;
- *WFUR* — estado em que o protocolo espera a resposta de seu usuário a um pedido de abertura de conexão da entidade remota.

Os testes feitos com o Abracadabra foram:

(1) A possibilidade do protocolo chegar à situação de conexão estabelecida, usando para a análise a fórmula ***S1 u S2***, onde ***S1*** foi declarado como verdadeiro e ***S2*** como o estado global objetivo constituído das instâncias ***Abracadabra A*** e ***B*** no estado parcial *OPEN*. O estado global inicial teria as instâncias ***Abracadabra A*** e ***B*** no estado parcial *CLOSED*.

(2) Teste de transferência de dados, tem o mesmo estado global inicial da análise nº (1). A fórmula de lógica temporal para esta análise foi: ***S1 -> (S2 u S3)***, onde ***S2*** foi declarado como verdadeiro, ***S1*** foi definido com as instâncias ***Abracadabra A*** e ***B*** no estado *OPEN*, sendo que ***A*** teria o pedido da primitiva ***DatReq*** (enviada pelo ***Usuário A***, solicitando envio de dado) e ***S3*** foi declarado

com o módulo ***Abracadabra A*** no estado *WFAK* e o módulo ***Abracadabra B*** no estado *OPEN* com a primitiva ***UnitInd*** (enviada pelo ***Meio DG***).

A verificação nº (1) foi feita de duas maneiras: a primeira sem que o **Verest+** acionasse os sistemas de lógica nebulosa (A e B) e a segunda, chamando-os normalmente. No primeiro caso, como mostra o trecho final do arquivo de descrição de resultados, apresentado a seguir (**Resultado11**), a análise levou 21 minutos e 9 segundos, percorrendo um caminho longo de 394 estados. No segundo caso, o estado objetivo foi atingido em apenas 12 segundos, após serem percorridos 17 estados globais. Veja o último trecho desse segundo resultado no arquivo **Resultado12** também a seguir.

Trecho do arquivo Resultado11 do Abracadabra

```
==========================================
Razao do final: Objetivo alcancado
==========================================
Conclusao:

Total de estados globais obtidos: 394
Tempo de verificacao: 0 horas, 21 minutos e 9 segundos
Resultado da formula -> FORMULA VERDADEIRA
Razao -> S1 tem ocorrido e S2 ocorreu
```

Trecho do arquivo Resultado12 do Abracadabra

```
==========================================
Razao do final: Objetivo alcancado
==========================================
Conclusao:

Total de estados globais obtidos: 17
Tempo de verificacao: 0 horas, 0 minutos e 12 segundos
Resultado da formula -> FORMULA VERDADEIRA
Razao -> S1 tem ocorrido e S2 ocorreu
```

O trecho do arquivo **Resultado2**, apresentado a seguir, mostra o resultado final da verificação nº (2). Pode-se ver, nesse arquivo, que o tempo de análise para provar a propriedade descrita na referida verificação foi de 6 minutos e 24 segundos com 179 estados globais desenvolvidos. Esse tempo foi bem maior do que a verificação nº (1) com os sistemas de lógica nebulosa, mas isso deveu-se pelo fato da verificação nº (2), apesar de ter sido feita também com os sistemas nebulosos, relacionar primitivas de serviço em seu estado global objetivo (***DatReq*** e ***UnitInd***) e a técnica de verificação usada pelo **Verest+** só manipular os estados parciais dos módulos do protocolo. Portanto, não há conhecimento, envolvendo mensagens e variáveis de protocolo nos sistemas de lógica nebulosa.

Trecho do arquivo Resultado2 do Abracadabra

```
==========================================
Razao do final: Objetivo alcancado
==========================================
Conclusao:

Total de estados globais obtidos: 179
Tempo de verificacao: 0 horas, 6 minutos e 24 segundos
Resultado da formula -> FORMULA VERDADEIRA
Razao -> S2 tem ocorrido e S3 ocorreu
```

5.4 Protocolo Multiponto-Multiponto

O serviço de comunicação do protocolo Multiponto-Multiponto (Vidal, 1994) oferece as seguintes facilidades aos seus usuários: meios de estabelecimento de uma conexão entre as estações participantes com fim de trocar ***SDUs*** (***service data unit***); meios para inserção de uma ou mais estações como participante em uma conexão já estabelecida de forma negociada; ou meios para liberação, negociada ou não, de uma ou mais estações; e meios para transferência de *SDUs* durante a conexão.

A arquitetura da especificação ESTELLE do Multiponto-Multiponto (MPMP) é constituída de 4 entidades de usuário (***UA***, ***UB***, ***UC*** e ***UD***), 4 entidades de enlace mulitponto-mulitponto (***ENL_MPMPa***, ***ENL_MPMPb***, ***ENL_MPMPc*** e ***ENL_MPMPd***) e uma entidade da camada física (***MACPHY***). A especificação do MPMP possui, ao todo, 10 estados parciais e 280 transições. Os estados parciais do serviço de estabelecimento de conexão do protocolo multiponto a multiponto são:

- *CLOSED* — A conexão multiponto está fechada e, no que diz respeito à estação, não há conferência em andamento.
- *ESTABLISH* — Foi enviado um pedido de conexão e a estação aguarda a confirmação de todas as possíveis estações participantes.
- *ESTABLISH RECEIVER* — Uma estação pode entrar nesse estado em duas condições: se após receber uma primitiva de requisição de conexão já estabelecida e que, por isso, a estação deverá entrar na conferência como receptora. O segundo motivo é o recebimento de uma ***PDU PLIST***, que contém a lista de participantes da conferência, sem a sua identificação. Isso configura que a estação não está na conferência. Em resposta a esse acontecimento é enviado um pedido de conexão para a estação Mestra e, em seguida, a estação vai para esse estado esperar a confirmação da conexão, para poder voltar a operar como receptora.
- *OPEN RECEIVER* — Nesse estado só é permitida a recepção de dados.
- *OPEN MASTER* — Nesse estado a estação é a Mestra da conferência, podendo transmitir dados via canal principal.

A técnica de verificação usada pelo **Verest+** foi criada para ser aplicada em especificações com duas entidades (local e remota). No entanto, apesar do protocolo MPMP possuir mais de duas entidades, é possível verificar diversas de suas propriedades fazendo-se os testes com duas entidades na definição do estado global objetivo.

O teste realizado com o MPMP é semelhante ao teste do HMS e à verificação nº (1) do Abracadabra, ou seja, foi inspecionada a sua abertura de conexão. Essa verificação foi feita em todos os protocolos, pois o sistema padrão empregado na comparação possui uma abertura de conexão negociada, depois uma fase de transmissão de dados e uma desconexão.

Para a análise da abertura de conexão do MPMP, as duas entidades ***ENL_MPMPa*** e ***ENL_MPMPb*** foram escolhidas arbitrariamente, pois o MPMP oferece um serviço de texto-conferência em que todas as entidades estejam envolvidas ou apenas duas delas. Sendo assim, tal situação de conexão entre duas entidades apenas constitui uma propriedade de interesse do protocolo e, logo, o **Verest+** pôde testá-la.

Novamente, como nos casos anteriores, foi usada a fórmula ***S1 u S2***, com ***S1*** declarado como verdadeiro e, ***S2*** sendo definido com a instância ***ENL_MPMPa*** no estado *OPEN_MASTER* e a instância ***ENL_MPMPb*** no estado *OPEN_REC*. Esses dois estados referem-se às duas instâncias alcançando a abertura de conexão a partir dos seus estados iniciais que se chamam *CLOSED*. A diferença entre eles é que, durante uma conferência, uma das estações deve ser a mestre para gerir o diálogo, enquanto as outras são consideradas como estações

receptoras secundárias.

O protocolo Multiponto-Multiponto foi usado como teste a fim de observar o comportamento do **Verest+** diante de um sistema sob estudo bem maior do que os demais protocolos submetidos a ele como teste também. Não era intenção verificar o protocolo completamente, pois o mesmo foi bem testado por Vidal (1994). Assim, foi feita apenas a análise já mencionada. O teste do MPMP foi bem sucedido, gerando 505 estados globais em 59 minutos e 43 segundos, conforme pode ser visto no trecho final do arquivo de descrição dos resultados do MPMP a seguir:

Trecho do arquivo de descrição dos resultados do MPMP

```
==========================================
Razao do final: Objetivo alcancado
==========================================
Conclusao:

Total de estados globais obtidos: 505
Tempo de verificacao: 0 horas, 59 minutos e 43 segundos
Resultado da formula -> FORMULA VERDADEIRA
Razao -> S1 tem ocorrido e S2 ocorreu
```

O tempo de 59 minutos e 43 segundos de análise do MPMP por si só não foi muito relevante para este teste, pois ao usar o exemplo do MPMP, procurava-se averiguar a eficácia do **Verest+** com sistemas bem grandes, uma vez que o mesmo protocolo Multiponto-Multiponto, por ser muito grande, nem sequer pôde ser analisado pela ferramenta ARP 2.3 (Farines, 1989) que rapidamente apresentou memória esgotada devido ao problema da explosão do espaço de estados. Em Vidal (1994), o MPMP foi verificado por meio de várias simulações que consumiram diversos dias.

5.5 Teste com o Sistema Padrão

O teste com o sistema padrão consistiu em usá-lo também como sistema sob estudo, ou seja, fazer a comparação do sistema padrão com ele mesmo. Esse teste foi idealizado, pois se trata de um caso base em que a técnica de verificação proposta (seguramente) deve fornecer a resposta correta dentro de um tempo muito curto.

Ao funcionar o teste com o sistema padrão, usado como próprio **SE**, isso estaria indicando que a técnica de verificação se comporta coerentemente. Uma vez que esse teste é o mais simples de todos os realizados, ou seja, é bem visível o comportamento do sistema padrão, foi decidido que seria uma boa oportunidade para introduzir um erro de especificação proposital no **SP** no sentido de observar a resposta do **Verest+** diante de tal situação. Resumindo, foram feitas as seguintes verificações com o sistema padrão usado como sistema sob estudo:

(1) Teste de abertura de conexão em que foi usada a fórmula ***S1 u S2***, onde ***S1*** foi declarado como verdadeiro e ***S2*** como o estado objetivo, tendo a entidade ***Iniciador*** no estado parcial *TRANSMITINDO* e a entidade ***Respondedor*** no estado *RECEBENDO* (veja figura 8).

(2) O mesmo teste de abertura de conexão, usando exatamente a mesma fórmula com o mesmo estado objetivo, porém introduziu-se um erro na especificação ESTELLE original. Tal erro consistiu em retirar o comando de envio da ***PDU CC*** (***connect confirm***) do ***Respondedor*** (veja figura 8).

No apêndice B, são mostrados todos os arquivos dos testes realizados com o sistema padrão. Na seção 9.1, apresenta-se a especificação ESTELLE, na qual se pode observar a linha da transição ***T2r*** onde foi colocado um comentário na cláusula *output MCEP.MediumReq (CC)* a fim de impedir que esse comando fosse executado, criando assim, o erro para a verificação nº (2). A verificação nº (2) foi feita sem o comentário nessa cláusula.

O resultado da primeira verificação com o sistema padrão pode ser observado nos arquivos de descrição dos resultados e seqüência de disparos no apêndice B (seções 9.2 e 9.3). No campo conclusão do trecho do arquivo de descrição dos resultados, observa-se que o teste levou 12 segundos para achar o objetivo após expandir 7 estados globais. O arquivo de seqüência de disparos registra o caminho percorrido pelo **Verest+** até ter alcançado o estado global objetivo.

Foi decidido apresentar a expansão do sistema padrão na seção 5.1 feita pelo programa ARP 2.3 (Farines, 1989), pois seria interessante comparar a análise do **SP** obtida através do **Verest+** com uma outra ferramenta já bem testada a fim de averiguar se o mesmo iria se comportar de maneira correta. De acordo com a tabela 3, o ARP mostra o objetivo no quinto estado global expandido (marcação **M4**), enquanto o **Verest+** achou o mesmo nó objetivo no sétimo estado global. Apesar da diferença, esse resultado na realidade é correto, pois na especificação ESTELLE existem duas transições a mais, ou seja, as transições da entidade ***Meio*** que não consta da especificação com redes de Petri, já que o modelo do meio de comunicação da figura 9, em rede de Petri, usa as transições dos modelos do ***Iniciador*** e do ***Respondedor***. Portanto, acompanhando o arquivo de seqüência de disparos das transições, caso fossem retiradas as transições **M1** e **M2** da instância de módulo ***Meio***, o **Verest+** também teria alcançado o objetivo no quinto estado global.

No campo razão do final do arquivo de descrição dos resultados da segunda verificação do **SP**, está registrado ***bloqueio***, enquanto o campo conclusão revela que não foi possível obter qualquer conclusão a respeito da fórmula testada, isto é, não pôde ser registrado se a mesma era verdadeira ou falsa, pois, sem que o objetivo tenha sido alcançado, nada pode ser dito em relação à fórmula. Esse mesmo bloqueio também encontra-se registrado no arquivo de seqüência de disparos das transições.

Acompanhando o modelo com redes de Petri da figura 8, vê-se claramente que o erro introduzido de propósito na especificação do **SP**, de fato o levaria a um bloqueio, como o **Verest+** corretamente o identificou, pois tal erro, retirada do envio da ***PDU CC*** (***connect confirm***), deixa o ***Iniciador*** eternamente no estado *ESPERA_CONF* aguardando a referida mensagem, enquanto o ***Respondedor***, igualmente ficaria para sempre no estado *RECEBENDO*, uma vez que o ***Iniciador*** estaria impedido de executar a desconexão.

6 CONCLUSÃO

Este artigo mostrou a implementação e a utilização de um verificador (**Verest+**) feito para testar especificações de protocolos descritos pela linguagem ESTELLE. O comportamento esperado para o **Verest+** seria apontar como verdadeiras, as propriedades efetivamente corretas (presentes) nos protocolos de comunicação. Foram testados os protocolos descritos na seção anterior com o intuito de comprovar a viabilidade do emprego do **Verest+**. Uma vez que tais protocolos (Fernandez e Pedroza, 1997; Courtiat, 1987) já haviam sido testados antes com os serviços descritos na seção anterior já provados, logo o **Verest+** deveria também constatar

a correção desses mesmos serviços, a não ser do teste de contra prova feito com a especificação do sistema padrão.

A implementação do **Verest+** foi importante para atestar a eficácia de se utilizar informação (heurística) a respeito do protocolo testado, durante a análise, pois conforme a verificação da abertura de conexão (nº 1), processada com o protocolo Abracadabra, pôde ser observado que a mesma análise feita sem informação levou muito mais tempo do que a outra com a informação extraída dos sistemas de lógica nebulosa ao ser feita a comparação com o sistema padrão. Outro teste bastante relevante nesse sentido, foi a verificação da transmissão de dados (nº 2) também com o Abracadabra, pois a mesma levou mais tempo a ser concluída pelo fato de ter levado em conta na fórmula da propriedade testada, objetos das instâncias de módulo não integrantes da informação manipulada pelos sistemas de lógica nebulosa, ou seja, mensagens trocadas.

O teste do **Verest+** com um protocolo grande como o MPMP demonstra que, mesmo usando-se uma técnica de verificação nova que acelera a análise, com base em buscas parciais, ainda assim o tempo de processamento de uma única propriedade levou quase uma hora, comprovando a importância de continuar perseguindo novos métodos de testes de protocolos de comunicação reais de tal forma a poder baixar cada vez esse tempo de análise.

A construção do **Verest+**, usando Lógica Nebulosa, surgiu da necessidade de se desenvolver uma nova técnica de verificação que viesse a suprir as desvantagens de ferramentas e métodos anteriores como, por exemplo, todas as técnicas baseadas em buscas exaustivas que apresentam o problema da explosão do espaço de estados ou análise combinatória. Dentro desse grupo podem ser citados os trabalhos de Bochmann (1978) e Hailpern (1980).

Existem outros trabalhos (Jard, 1988; West, 1992) que usam simulações ao invés de buscas exaustivas, evitando assim o problema da explosão combinatorial, todavia tais ferramentas não usam informação a respeito do sistema testado, podendo ficar muito tempo sem chegar a qualquer conclusão a respeito da correção do mesmo.

7 REFERÊNCIAS BIBLIOGRÁFICAS

Bernardo Filho, O. (1999). Verificação de Protocolos de Comunicação com Lógica Nebulosa. Tese de Doutorado, Programa de Engenharia Elétrica, COPPE/UFRJ, Rio de Janeiro RJ.

Bernardo Filho, O., A. C. P. Pedroza e J. L. S. Leão (1992). O Verificador de Um Sistema de Auxílio ao Projeto de Protocolos. *Anais do 9º Congresso Brasileiro de Automática*, Vitória ES.

Bernardo Filho, O., A. C. P. Pedroza e J. L. S. Leão (1998a). Uma Técnica com Lógica Nebulosa para Verificação de Sistemas Distribuídos. *Anais do 12º Congresso Brasileiro de Automática*, Uberlândia MG.

Bernardo Filho, O., A. C. P. Pedroza, J. L. S. Leão e M. P. Fernandez (1998b). Uma Técnica com Lógica Nebulosa para Verificação de um Protocolo de Gerência Hierárquica de Rede. *Anais do 16º Simpósio Brasileiro de Redes de Computadores*, Rio de Janeiro RJ.

Bochmann, G. V. (1978). Finite state description of communication protocols, *Computer Networks*, 2.

Courtiat, J. P. (1987). Contribution a la Description Formelle de Protocoles. These du Grade de Docteur d'Etate (sciences), A L'Universite Paul Sabatier de Toulouse.

Dubois, D. e H. Prade (1980). *Fuzzy Sets and Systems, Theory and Applications,* Academic Press, Orlando.

Farines, J.-M. (1989). ARP 2.3 — Analisador de Redes de Petri, Programa desenvolvido a partir de uma tese de mestrado, Laboratório de Controle e Microinformática, EEL/UFSC, Florianópolis SC.

Fernandez, M. P. e A. C. P. Pedroza (1997). Sistema de Gerenciamento Hierárquico para Grandes Redes de Telecomunicações. *Anais do 15º Simpósio Brasileiro de Telecomunicações*.

Hailpern, B. T. (1980). Verifying network protocols using temporal logic, *Proceedings NBS Trends and Applications Conf.*

International Organization for Standardization (ISO) (1989). Information processing systems - open systems interconnections - ESTELLE - a formal description technique based on an extended state transition model. ISO 9074.

Jard, C. (1988). Valider les protocoles en simulant, *Colloque Francophone Sur L'Ingénierie des Protocoles - CFIP'88*, Bordeaux, setembro.

Lee, C. C. (1990). Fuzzy Logic in Control Systems: Fuzzy Logic Controller, parts I & II, *IEEE Transactions on Systems*, Man and Cybernetics, vol. 20, nº 2.

Manna, Z. e A. Pnueli (1981). Verification of Concurrent Programs: Temporal Proof Principles. *Proceedings of Workshop on Logics of Programs*, Lectures Notes in Computer Science, Springer-Verlag.

Mendel, J. M. (1995). Fuzzy Logic Systems for Engineering: A Tutorial. *Proceedings of the IEEE*, vol. 83, nº 3.

Oliveira Jr., R. C. (1991). Um Compilador para a linguagem ESTELLE. Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, Rio de Janeiro RJ.

Pedrycz, W. e F. Gomide (1998). *An Introduction to Fuzzy Sets: Analysis and Design (Complex Adaptive Systems)*. MIT Press.

Rescher, N. e A. Urquhart (1971). *Temporal Logic*, Springer-Verlag Wien New York.

Rich, E. e K. Knight (1991). *Artificial Intelligence*, Second Edition, International Edition, McGraw-Hill Inc.

Terano, T., K. Asai, e M. Sugeno, (1987). *Fuzzy Theory and its Applications*, Academic Press, Boston.

Vidal, M. T. V. L. (1994). Especificação Formal, Verificação e Implementação de um Protocolo de Comunicação Multiponto-a-Multiponto e uma Aplicação de Texto-Conferência. Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, Rio de Janeiro RJ.

West, C. H. (1992). Protocol validation - principles and applications, *Computer Networks and ISDN Systems*,

North-Holland.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*.

Zadeh, L. A. (1978). Fuzzy Sets as a Basis for a Theory of Possibility, *Fuzzy Sets and Systems 1*.

Zimmermann, H. J. (1985). *Fuzzy Set Theory - and Its Applications*, Kluwer-Nijhoff Publishing.

8 APÊNDICE A –– PSEUDO-CÓDIGO DO ALGORITMO A*

```
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         }
5   {de x ao nó meta.                                                    }

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

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

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

21  INÍCIO
22  Ler (NÓ_RAIZ);
23  AcheiObjetivo <- FALSO;
24  primFECHADO <- nil;
25  primABERTO <- NÓ_RAIZ;
26  NÓ_RAIZ^.g <- 0;
27  NÓ_RAIZ^.próximo <- nil;
28  NÓ_RAIZ^.ptrSuc <- nil;
29  REPETIR
30  MELHOR_NÓ <- primABERTO;
31  primABERTO <- primABERTO^.próximo;
32  MELHOR_NÓ^.próximo <- primFECHADO;
33  primFECHADO <- MELHOR_NÓ;
34  AcheiObjetivo <- É_Objetivo (MELHOR_NÓ);
35  SE NÃO AcheiObjetivo
36  ENTÃO
37  SUCESSORES <- GerarSuc (MELHOR_NÓ);
38  ENQUANTO SUCESSORES # nil FAÇA
39  TRATA_SUCESSORES (SUCESSORES^.pSuc);
40  SUCESSORES <- SUCESSORES^.próximo;
41  FIMENQUANTO
42  FIMSE
43  ATÉ AcheiObjetivo OU (primABERTO = nil);
44  FIM {A*}

45  SUBALGORITMO TRATA_SUCESSORES (Suc : PONTEIRO
    PARA NÓ);

46  VARIÁVEIS:
47  Estou_em_aberto, Estou_em_fechado : BOOLEANO;

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

9 APÊNDICE B –– ARQUIVOS DOS TESTES COM O SISTEMA PADRÃO

9.1 Especificação ESTELLE

```
specification SISTEMApadrao systemactivity;

default individual queue; timescale seconds;

(*-------------------------------------------------*)

type
   DataType = (CR,CC,DT,DR);
(*-------------------------------------------------*)

channel MEDIUMINT (USER,PROVIDER);
  by USER:
    MediumReq (sdu:DataType);
  by PROVIDER:
    MediumInd (sdu:DataType);
(*-------------------------------------------------*)

module SPType activity;
  ip MCEP: MEDIUMINT (USER);
end;

module MEDIUMType activity;
  ip MCEP: array [0..1] of MEDIUMINT (PROVIDER);
end;
(*-------------------------------------------------*)

body InicBODY for SPType;

state
    FECHADOi, ESPERAconf, TRANSMITINDO, DESCONECTAR;

initialize to FECHADOi begin end;

trans
    from FECHADOi to ESPERAconf
       name T1i: {? ConnectReq}
       begin
         output MCEP.MediumReq (CR)
       end;
    from ESPERAconf to TRANSMITINDO
      when MCEP.MediumInd
       provided sdu = CC
        name T2i:
        begin {! ConnectConf}
        end;
    from TRANSMITINDO to DESCONECTAR
       name T3i: {? DataReq}
       begin
```

```
        output MCEP.MediumReq (DT)
      end;
    from DESCONECTAR to FECHADOi
      name T4i: {? DisconnectReq}
      begin
        output MCEP.MediumReq (DR)
      end;
end; (* Iniciador *)
(*-------------------------------------------------*)

body RespBODY for SPType;

state
    FECHADOr, ESPERArespUSU, RECEBENDO;

initialize to FECHADOr begin end;

trans
    from FECHADOr to ESPERArespUSU
      when MCEP.MediumInd
       provided sdu = CR
        name T1r:
        begin {! ConnectInd}
        end;
    from ESPERArespUSU to RECEBENDO
      name T2r: {? ConnectResp}
      begin
        {output MCEP.MediumReq (CC)}
  {este comentario na clausula output testa um erro}
      end;
    from RECEBENDO to RECEBENDO
      when MCEP.MediumInd
       provided sdu = DT
        name T3r:
        begin {! DataInd}
        end;
    from RECEBENDO to FECHADOr
      when MCEP.MediumInd
       provided sdu = DR
        name T4r:
        begin {! DisconnectInd}
        end;
end; (* Respondedor *)
(*-------------------------------------------------*)

body MEDIUMBODY for MEDIUMType;

state
    IDLE;

initialize to IDLE begin end;

trans
    from IDLE to IDLE
      when MCEP[0].MediumReq
        name M1:
        begin
          output MCEP[1].MediumInd (sdu)
        end;
      when MCEP[1].MediumReq
        name M2:
        begin
          output MCEP[0].MediumInd (sdu)
        end;
end; (* MEDIUM *)
(*-------------------------------------------------*)

modvar
     Iniciador,Respondedor : SPType;
     Meio                  : MEDIUMType;

initialize

begin
  init Iniciador with InicBODY;
  init Respondedor with RespBODY;
  init Meio with MEDIUMBODY;
  connect Iniciador.MCEP to Meio.MCEP[0];
  connect Respondedor.MCEP to Meio.MCEP[1];
end;

end. (* SISTEMApadrao *)
```

9.2 Descrição dos Resultados

Verificação (1)

```
==================================================
Razao do final: Objetivo alcancado
==================================================
Conclusao:

Total de estados globais obtidos: 7
Tempo de verificacao: 0 horas, 0 minutos e 12 segundos
Resultado da formula -> FORMULA VERDADEIRA
Razao -> S1 tem ocorrido e S2 ocorreu
```

Verificação (2)

```
==================================================
Razao do final: Bloqueio
==================================================
Conclusao:

Total de estados globais obtidos: 5
Tempo de verificacao: 0 horas, 0 minutos e 36 segundos
Conclusao nao foi possivel
```

9.3 Seqüência de Disparos

Verificação (1)

```
Estado inicial = (1) -{1}T1I-> (2) -{3}M1-> (3) -{2}T1R-> (4)-{2}T2R-> (5)
-{3}M2-> (6) -{1}T2I-> (7)
***** OBJETIVO ALCANCADO *****
```

Verificação (2)

```
Estado inicial = (1) -{1}T1I-> (2) -{3}M1-> (3) -{2}T1R-> (4) -{2}T2R-> (5)
***** BLOQUEIO *****

***** OBJETIVO NUNCA ALCANCADO *****
```