UMA FERRAMENTA PARA VERIFICAÇÃO DE PROTOCOLOS DE COMUNICAÇÃO: IMPLEMENTAÇÃO E EXPERIÊNCIAS

Orlando Bernardo Filho[1,2], 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/aloysio@gta.ufrj.br/leao@gta.ufrj.br
(3) Departamento de Eletrônica/EE-UFRJ

Resumo

Este trabalho apresenta a implementação e algumas das experiências realizadas com uma ferramenta de verificação para auxiliar na correção de erros de protocolos de comunicação. Tal ferramenta, chamada de **Verest+**, foi construída valendo-se de conceitos da Lógica Nebulosa e da Lógica Temporal. As experiências, aqui relatadas, referem-se a um protocolo de gerência de rede e ao protocolo Abracadabra.

Abstract

This paper shows some experiences and the implementation of a verification tool to aid the proof of communication protocols. Such tool, named **Verest+**, was designed with Fuzzy Logic and Temporal Logic concepts. The experiences, shown here, were done with a network management protocol and the Abracadabra protocol.

Palavras chaves: métodos formais, verificação de protocolos, sistema de lógica nebulosa.

1 INTRODUÇÃO

As redes de computadores evoluiram muito desde a década de 70 graças, inicialmente, ao fabuloso avanço da tecnologia de ***hardware***, pois hoje em dia dispõe-se de um conjunto de componentes eletrônicos bem mais rápidos e eficientes, que além disso incorporaram muito mais funções devido à integração em larga escala dos circuitos digitais. Acompanhando essa evolução do ***hardware***, houve também um grande desenvolvimento das técnicas e métodos de projeto e análise dos recursos de ***software*** para as redes de computadores. Tal desenvolvimento teve origem na padronização das arquiteturas dos protocolos de comunicação, seguindo-se da criação dos métodos formais que viabilizaram o projeto de redes de computadores de maneira mais segura e melhor documentada.

Os métodos formais que foram criados na década de 80 alcançaram parte de seus objetivos ao disciplinar o projeto das redes de computadores, mas algumas das técnicas propostas nessa ocasião foram ousadas demais, ao pretenderem produzir de maneira automática quase todo um sistema e, por outro lado, algumas dessas técnicas que efetivamente atingiram um bom nível de automação, apresentavam uma grande dificuldade para o projetista usá-las tamanha era a complexidade dos métodos envolvidos.

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. Tal enfoque tem como objetivo conduzir essas ferramentas para o setor industrial, pois até o momento, a maior parte dos métodos formais criados se limitaram ao meio acadêmico e científico.

O trabalho aqui descrito procurou se guiar por esses princípios atuais, uma vez que a ferramenta para verificação de protocolos de comunicação emprega um Sistema de Lógica Nebulosa que otimiza a busca no grafo de execução do protocolo sob análise, acelerando a obtenção dos objetivos. O Sistema de Lógica Nebulosa usado, na realidade, implementa uma heurística por meio de suas regras de inferência o que é mais fácil para o usuário, já que o mesmo tem apenas que fornecer alguns parâmetros para o verificador ao invés de toda uma definição do seu conhecimento de protocolo para montar uma heurísitca.

Este artigo encontra-se dividido em quatro seções além das conclusões e referências bibliográficas. Na seção seguinte, indroduzem-se os principais conceitos da Lógica Temporal, da Lógica Nebulosa e da técnica de busca heurística empregados na construção da ferramenta de verificação **Verest+**. Na terceira seção, mostra-se a implementação do **Verest+**, onde podem ser vistas as descrições de todos os seus módulos. Apresenta-se, na quarta seção, os resultados das experiências realizadas com os protocolos de Gerência Hierárquica de rede e Abracadabra.

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. O **Verest+** é a aplicação de uma técnica de verificação de protocolo proposta em [1]. Os protocolos a serem testados pelo **Verest+** devem ser especificados em ESTELLE [2], pois o mesmo trata-se de uma evolução de um outro verificador para ESTELLE [3] que não contava com uma heurística construída por meio de um sistema de lógica nebulosa.

O **Verest+** usa as fórmulas da Lógica Temporal para descrever as propriedades a serem testadas. A Lógica Temporal [4,5] 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 mais usados no **Verest+**.

- Operador ***sempre*** ou ***doravante*** (denotado pelo símbolo $\Box$) cuja fórmula formada com uma condição φ ($\Box\varphi$) que pode ser, inclusive, outra fórmula da lógica temporal, é verdadeira num estado s_i, se e somente se, φ for verdade em todos os estados s_j acessíveis a partir de s_i (incluindo o próprio s_i).

- Operador ***eventualmente*** ou ***alguma vez*** (denotado pelo símbolo ◊) que ao ser aplicado a uma condição φ ($\lozenge\varphi$), 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*** (denotado pelo símbolo u), único operador binário dos três. Uma fórmula básica com duas condições φ e ψ ($\varphi\, u\, \psi$) 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.

As fórmulas a serem validadas são aquelas que especificam as propriedades mais comumente usadas para verificar sistemas concorrentes, segundo uma classificação proposta por MANNA et alia [5]. A tabela 1 a seguir mostra as fórmulas usadas pelo **Verest+**.

Tabela 1: Fórmulas da lógica temporal que podem ser usadas na análise do **SE**.

NÚMERO	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)
5	S1 -> &(S2 u S3)	S1 implica eventualmente (S2 até S3)
6	S1 -> [](S2 u S3)	S1 implica sempre (S2 até S3)
7	~(S1 u S2)	Não (S1 até S2)
8	[](S1 -> S2)	Sempre (S1 implica S2)
9	&(S1 -> S2)	Eventualmente (S1 implica S2)
10	S1 -> [](S2 -> S3)	S1 implica sempre (S2 implica S3)

A Lógica Nebulosa [6,7,8,9] é usada no **Verest+** para compor um sistema nebuloso responsável pela heurística de busca no grafo de execução do protocolo sob análise. A Lógica Nebulosa é uma aplicação da teoria dos conjuntos nebulosos. 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.

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***.

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

O ***fuzificador*** mapeia um valor preciso (***crisp***) $\boldsymbol{x} \in \boldsymbol{U}$ para um conjunto nebuloso $\tilde{\mathbf{A}}$ em $\boldsymbol{U}$. Um parâmetro preciso pode ser ***fuzificado***, seguindo um método simples, trapezoidal, triangular ou gaussiano. O método mais usado é o simples que nada mais é do que criar o conjunto nebuloso $\tilde{\mathbf{A}}$ com sua função de pertinência [6,7,8,9] assumindo valor **1** no ponto de $\boldsymbol{U}$ em que o dado parâmetro tem a sua definição e, nos demais pontos, o valor da pertinência seria **0**. A ***desfuzificação*** pode ser feita através do valor máximo, da média de valores máximos ou por meio do cálculo do centróide.

Um sistema de lógica nebulosa típico será composto de várias regras de inferência, manipulando termos de algumas variáveis lingüísticas como no exemplo exposto a seguir que trata-se, na realidade, das regras do sistema de lógica nebulosa implementado no **Verest+**.

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

As regras apresentadas anteriormente modelam um raciocínio nebuloso com o intuito de ordenar a fila de estados globais *ABERTOS* do protocolo sob análise. As variáveis lingüísticas empregadas nesse exemplo seriam:

- *T(**semelhança**)={diferente,pouco_semelhante,semelhante,muito_semelhante,igual}*
- *T(**custo**)={muito_baixo,baixo,alto,muito_alto}*
- *T(**posição**)={início,perto_do_início,meio,perto_do_fim,fim}*

O termo $\{\eta_s,\sigma_s\}$ representa dois estados globais em um dado momento da verificação, onde η_s trata-se de um estado global do protocolo sob análise e σ_s representa um estado global de um outro sistema pré-conhecido, a ser chamado de sistema padrão [1] o qual é comparado com o mencionado protocolo sob teste. O termo $\tilde{C}o(\sigma_s)$ refere-se ao custo nebuloso do estado global σ_s ao seu objetivo, enquanto $\tilde{P}(\eta_s)$ é a posição nebulosa obtida como resposta que o estado global η_s deve ocupar na fila de *ABERTOS*.

Os sistemas axiomáticos, construídos com lógica clássica, possuem um conjunto de implicações ou premissas (axiomas) descrevendo o conhecimento sobre um determinado assunto. Ao ser feita uma pergunta a esse conjunto de axiomas básicos, pode-se deduzir uma fórmula como resposta do sistema através da aplicação de regras de inferência que atuam sobre as fórmulas constituintes dos referidos axiomas. Uma das regras de inferência utilizadas na lógica clássica é a regra ***modus ponens***. Na lógica nebulosa é necessário estender a definição dessa regra de inferência que passa, nesse caso, a ser chamada de ***modus ponens generalizado***, pois os dados da lógica nebulosa são aproximações de uma realidade. Portanto, a regra ***modus ponens generalizado*** ficaria da seguinte forma:

$$\begin{array}{c} A \supset B \\ \underline{A'\quad\quad} \\ B' \end{array} \tag{1}$$

A expressão (1) 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, como no sistema mostrado anteriormente da ordenação da fila de *ABERTOS*, existem dois conjuntos nebulosos de entrada para comparar com cada um dos antecedentes das regras. No caso especial em que tais entradas fossem iguais aos termos das variáveis lingüísticas ***semelhança*** e ***custo***, de alguma regra, então a resposta corresponderia ao conseqüente dessa mesma regra. Entretanto, o caso geral corresponde à situação em que tais entradas não são iguais aos antecedentes de nenhuma das regras e sim uma boa aproximação dos antecedentes de alguma regra e uma má aproximação dos antecedentes das demais regras.

A aplicação da inferência ***modus ponens generalizado*** ao caso de várias regras é feita por intermédio de um cálculo [10] 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 o exemplo visto anteriormente:

(1) Sejam $\tilde{S}$ e $\tilde{C}$ dois conjuntos nebulosos representando uma semelhança entre os estados globais $\{\eta_s,\sigma_s\}$ e um custo do estado global σ_s ao objetivo. Sejam também $\mu_{\tilde{S}}$ e $\mu_{\tilde{C}}$ as funções de pertinência de $\tilde{S}$ e $\tilde{C}$ respectivamente.

*(2) Para cada regra **Rn** Faça:*

(2.1) Achar a função mínima entre a função de pertinência do antecedente Semelhança de $\{\eta_s,\sigma_s\}$ e $\mu_{\tilde{S}}$, nomeando-a de $\mathbf{f_{m\acute{i}nA1}}$;

(2.2) Achar a função mínima entre a função de pertinência do antecedente $\tilde{C}o(\sigma_s)$ e $\mu_{\tilde{C}}$, nomeando-a

de $\mathbf{f_{mínA2}}$*;*

(2.3) α_n *= mínimo (supremo (*$\mathbf{f_{mínA1}}$*), supremo (*$\mathbf{f_{mínA2}}$*));*

(2.4) Achar a função mínima entre a função constante $\mathbf{f = \alpha_n}$ *e o conseqüente de* ***Rn****, nomeando-a de* $\mathbf{f_{mínCn}}$*;*

(3) Fim do Para .... Faça

(4) Achar a função máxima entre todas as funções $\mathbf{f_{mínCn}}$ *para* ***n*** *variando de* ***1*** *à quantidade de regras.*

No exemplo considerado, as duas entradas $\tilde{S}$ e $\tilde{C}$ seriam resultados de uma ***fuzificação*** de valores precisos da semelhança e do custo. A resposta para essas duas entradas seria um conjunto nebuloso com universo de discurso da variável lingüística posição e com função de pertinência obtida no passo (4). Procedendo-se então com a ***desfuzificação***, de tal resposta, obter-se-ia finalmente um valor preciso da posição do estado global η_s com características de semelhança e custo $\tilde{S}$ e $\tilde{C}$ respectivamente.

A técnica de verificação usada na construção do **Verest+** vista em [1] 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*** [11], modificado para operar com variáveis lingüísticas da Lógica Nebulosa, passando a ser chamado de **Ã***. No algoritmo **A***, apresentado na figura 1 a seguir, 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). Portanto, a diferença entre o algoritmo **A*** e **Ã*** encontra-se embutida no procedimento ***TRATA_SUCESSOR***.

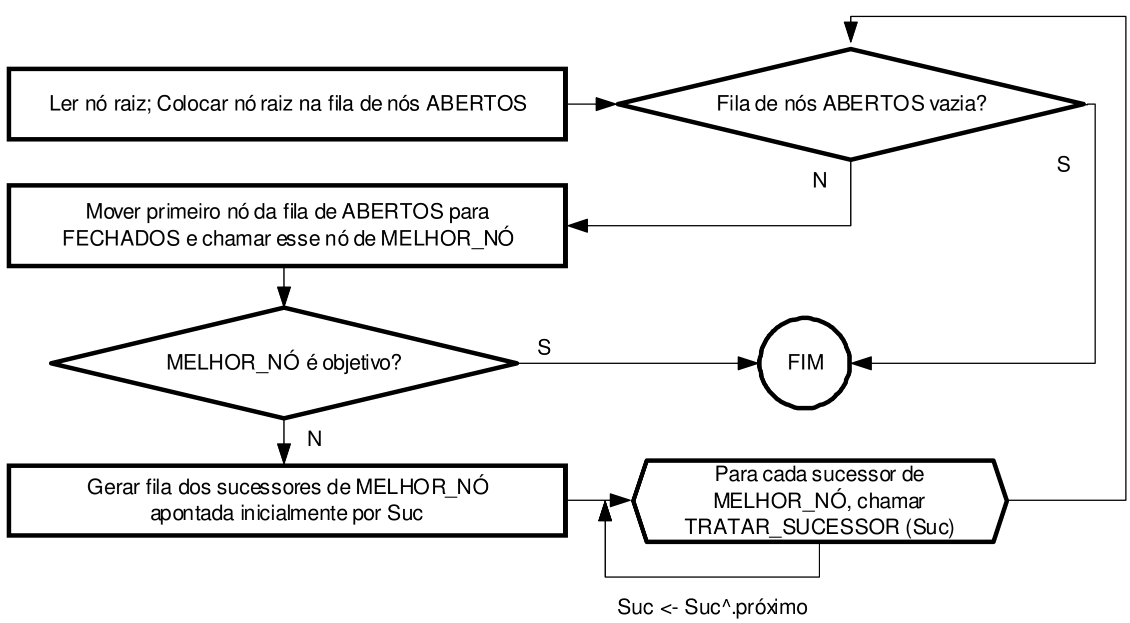


Figura 1: Fluxograma do algoritmo **A***.

A técnica de verificação é aplicada em um determinado protocolo de comunicação, doravante chamado de Sistema sob Estudo (**SE**), o qual deverá ser especificado em

ESTELLE. A especificação de **SE** deverá ser comparada à especificação de um outro sistema, denominado de Sistema Padrão (**SP**), considerado como um "bom" paradigma para **SE**. A partir do estabelecimento das semelhanças entre os objetos do **SE** e do **SP**, feito pelo usuário de forma nebulosa, começa o emprego da técnica de verificação. Em outras palavras, tem-se:

- **E** = $\{e_1, e_2, ..., e_{n1}\}$ é o conjunto (clássico) dos estados parciais de **SE**;
- **X** = $\{x_1, x_2, ..., x_{n2}\}$ é o conjunto (clássico) dos estados parciais de **SP**;

A partir de então, define-se a relação nebulosa $\tilde{R}_{EX}(e,x)$ 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 do **SE**.

Antes de serem executadas as regras do sistema de lógica nebulosa que conduz à ordenação da fila de *ABERTOS* (SLN B), deve-se processar um outro sistema de lógica nebulosa (SLN A) que obtenha a semelhança entre os estados globais ou nós do **SE** e do **SP**. A obtenção dessa semelhança é necessário, pois o usuário fornece a semelhança entre os estados parciais do **SE** e do **SP**, mas os estados globais formados por estados parciais do **SE** podem ter equivalentes em **SP** que não formam estados globais no mesmo **SP**. A resposta do SLN A constitui uma das entradas do SLN B.

3 IMPLEMENTAÇÃO DO Verest+

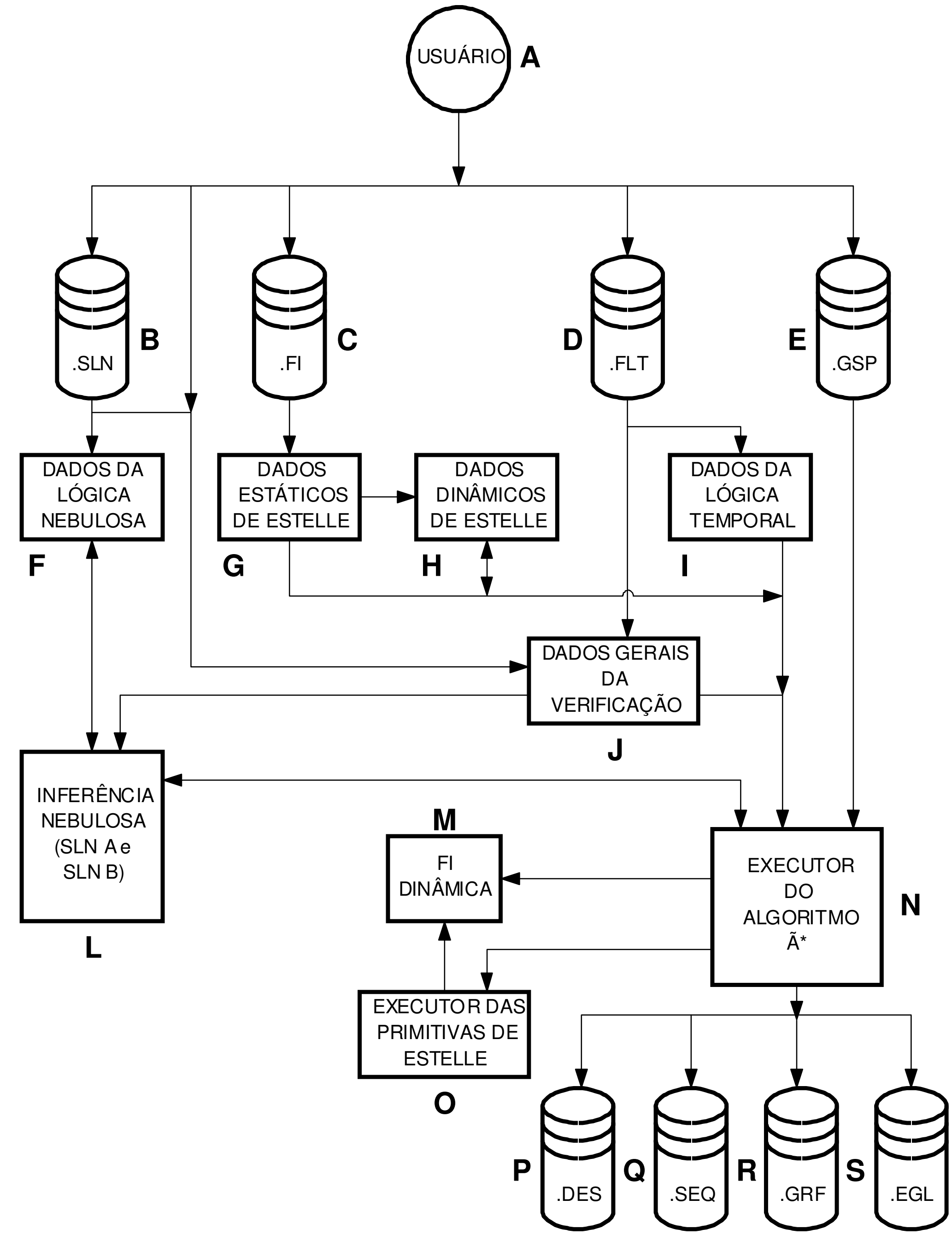
A figura 2, a seguir, apresenta a arquitetura do **Verest+**, implementado com base na técnica descrita [1]. O **Verest+** foi desenvolvido com a linguagem MODULA-2, tendo uma versão para DOS e outra para Linux, possuindo ao todo 120 módulos com aproximadamente 30 000 linhas de código. O bloco ***A*** representa o seu usuário que tem como função elaborar os arquivos de entrada para o verificador (blocos ***B***, ***C***, ***D*** e ***E***), além de fornecer ao mesmo, a identificação das instâncias de módulo, envolvidas na análise dos sistemas de lógica nebulosa. Tal informação fica armazenada no bloco ***J*** (dados gerais da verificação).

O arquivo do bloco ***B***, que tem em seu nome a extensão ***.SLN*** (Sistema de Lógica Nebulosa), traz a definição dos dados para o processamento da inferência nebulosa através da execução do SLN A e B. O bloco ***C*** representa o arquivo da forma intermediária estática gerado pelo compilador ESTELE [12]. Tal arquivo, cujo nome tem extensão ***.FI***, traz a definição de todos os dados que permanecem constantes durante a verificação da especificação ESTELLE, que descreve o sistema sob estudo, como, por exemplo, os identificadores das transições dos módulos do **SE**.

O arquivo caracterizado pelo bloco ***D***, cujo nome possui a extensão ***.FLT***, contém a definição, por parte do usuário, 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, o estado global objetivo etc. O último arquivo de entrada, representado pelo bloco ***E***, tem a extensão ***.GSP*** em seu nome 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 ***.SLN***, 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) registram durante a verificação, as informações variáveis do sistema sob estudo como, por exemplo, os estados parciais que possuem fichas, o conteúdo das filas associadas aos pontos de interação, entre outros.



<u>**Figura 2**</u>: Estrutura modular do verificador **Verest+**.

O arquivo ***.FLT***, tem as suas informações armazenadas nos blocos ***J*** e ***I*** (dados da lógica temporal). O bloco ***L*** representa o motor de inferência dos Sistemas de Lógica Temporal 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 cujos nomes possuem como extensão, respectivamente, ***.DES*** (arquivo de descrição dos resultados), ***.SEQ*** (arquivo que registra a seqüência de disparo de transições do caminho analisado), ***.GRF*** (arquivo que possui o grafo parcial do sistema sob estudo, obtido na análise) e ***.EGL*** (arquivo que possui uma descrição textual dos estados globais do **SE** expandidos durante a análise).

4 EXPERIÊNCIAS E RESULTADOS

O sistema padrão (**SP**) utilizado na comparação com os três protocolos verificados que serviram de testes para o **Verest+**, trata-se de um protocolo simples, envolvendo duas entidades comunicantes. Esse protocolo faz a abertura de uma conexão, transmite dados e então, encerra a conexão.

4.1 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 [13], tendo sido baseado em um modelo hierárquico de gerência. Pode-se ver em [13] o problema da gerência de uma grande rede colocado com bastante detalhe, aqui, apenas será apresentado o teste com um de seus serviços. 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 3 a seguir mostra a arquitetura da especificação ESTELLE do protocolo HMS.

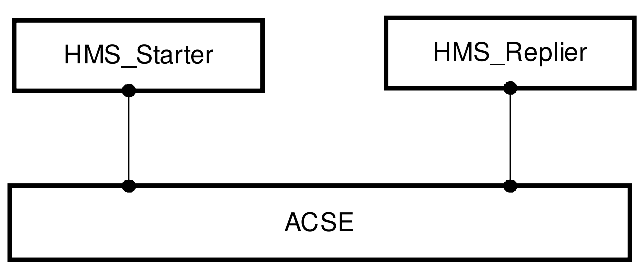


Figura 3: Arquitetura da especificação ESTELLE do protocolo HMS.

Na figura 3, pode-se ver as 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. Já o módulo

HMS_Replier possui os estados ***IDLE***, ***CONNECTED*** e ***WaitDISCONNECT***. Quando o módulo ***HMS_Replier*** entra no estado ***WaitDISCONNECT***, significa que o mesmo já recebeu 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 ***WaitTABLEPOLICY*** e ***WaitDISCONNECT***, respectivamente.

A verificação do objetivo η_o do protocolo HMS é apresentada no arquivo de descrição dos resultados (HMS.DES). 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*** (vide 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.

<u>Trecho do arquivo HMS.DES</u>

```
=================================================================
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
```

A definição da fórmula ***S1 u S2*** e do objetivo, podem ser vistos no arquivo HMS.FLT. Também pode ser visto, o arquivo HMS.SLN que contém a definição da relação nebulosa $\tilde{R}_{EX}(e,x)$. O arquivo de seqüência de disparos HMS.SEQ mostra o caminho percorrido pelo **Verest+** que o levou da raiz ao objetivo. Veja todos esses arquivos a seguir.

<u>Arquivo HMS.FLT</u>

```
FORMULA: S1 u S2
=================================================================

Operando S1:

SimNOT DESATIVADO
QuantInstModulo *

Operando S2:

SimNOT DESATIVADO
QuantInstModulo >= 2

modulo 1 ------------------------------------------------------
corpo: HMSStarterBODY
cabecalho: HMSTYPE
estado: WaitTABLEPOLICY
filhos: *
filas: *
expressoes: *
modulo 2 ------------------------------------------------------
corpo: HMSReplierBODY
```

cabecalho: HMSTYPE
estado: WaitDISCONNECT
*filhos: **
*filas: **
*expressoes: **

===
OBJETIVO: estado operando 2
===
*ExpNaoUsadas: **

No arquivo ***.FLT***, o usuário deverá especificar (no lugar de ***S1 u S2***) uma das dez fórmulas da lógica temporal possíveis de serem testadas pelo **Verest+**. Tais fórmulas devem ser escritas conforme aparecem na tabela 1. Seguindo-se a declaração da fórmula a ser testada passa-se para a definição dos estados globais operandos ***Si*** (i = 1, 2 e 3) que representam um determinado contexto de computação da especificação ESTELLE. Esses operandos são constituídos de:

(a) A quantidade total de instâncias de módulo que pode ser estabelecida pelo usuário como: um valor arbitrário (*); um valor fixo α (= α); um valor maior ou igual a uma certa constante α ($\geq \alpha$).
Se o usuário declarar valor arbitrário, será impossível definir qualquer objeto de qualquer instância de módulo presente no contexto da especificação, uma vez que não será conhecido o total de instâncias existentes.

(b) A definição de alguns objetos de ESTELLE do total de instâncias de módulo dado por α (item (a)). A seção de definição dos objetos de cada módulo é feita pela palavra chave *modulo* seguida do número da instância e depois por alguns caracteres "-" (sinal de menos). Os objetos definíveis para cada uma das instâncias são:

- Identificador de corpo de módulo (palavra chave *corpo:*).
- Identificador de cabeçalho de módulo (palavra chave *cabecalho:*).
- Identificador de estado de módulo (palavra chave *estado:*).
- Instâncias de módulo filhas que são identificadas pelo número que aparece seguindo a palavra chave *modulo* na definição dos objetos de cada instância. Esse número é a referência de cada instância de módulo m_j (j = 1, 2,…, α) de um estado global operando ***Si***.
- Conteúdo das filas FIFO dos pontos de interação internos (referentes ao identificador de corpo de módulo já definido) ou externos (referentes ao identificador de cabeçalho de módulo já definido).
- Predicados usando os símbolos = (igual), # (diferente), > (maior), < (menor), >= (maior ou igual), e <= (menor ou igual) que relacionam valores de variáveis, expressões e parâmetros de módulo referentes ao identificador de corpo já definido.

(c) Um campo booleano chamado ***SimNOT*** que determina se o estado global operando ***Si***, definido pelos itens (a) e (b), deve assumir valor-verdade **v** (verdadeiro) no caso de ser verificado num certo contexto de computação C_x, ou se deve assumir valor-verdade **f** (falso) em C_x. Em outras palavras, esse campo indica se deve ser verificado ***Si*** (sintaxe → *SimNOT DESATIVADO*) ou, **~ *Si*** (sintaxe → *SimNOT ATIVADO*).

Arquivo HMS.SLN

DEFINICOES DO SISTEMA PADRAO:

CUSTO DO MAIOR CAMINHO: 7

ESTADOS DO CORPO LOCAL (1): IniciadorBODY

FechadoI
EsperaConf
Transmitindo
Desconectar

ESTADOS DO CORPO REMOTO (2): RespondedorBODY

FechadoR
EsperaRespUsu
Recebendo
===

DEFINICAO DA RELACAO NEBULOSA:

(HMSStarterBODY.IDLE,IniciadorBODY.FechadoI) = 1
(HMSReplierBODY.IDLE,RespondedorBODY.FechadoR) = 1
(HMSStarterBODY.WaitCONNECTION,IniciadorBODY.EsperaConf) = 1
(HMSStarterBODY.WaitTABLEPOLICY,IniciadorBODY.Transmitindo) = 0.9
(HMSReplierBODY.CONNECTED,RespondedorBODY.Recebendo) = 0.9
(HMSReplierBODY.WaitDISCONNECT,RespondedorBODY.Recebendo) = 0.7

FIM

Como pode ser observado acima, a definição do arquivo ***.SLN*** (Sistema de Lógica Nebulosa) começa com a declaração "*DEFINICOES DO SISTEMA PADRAO:*", sendo seguido pelo registro do custo do maior caminho do grafo do sistema padrão, ou seja, o maior número de disparos de transições que separam dois estados globais do **SP**. Esse parâmetro é usado para compor o universo de discurso da variável lingüística ***custo***, utilizada no Sistema de Lógica Nebulosa B, que ordena a fila de nós *ABERTOS*.

Depois do estabelecimento do custo do maior caminho do **SP**, o arquivo ***.SLN*** passa a mostrar a identificação dos estados parciais de **SP**, envolvidos na análise. São declarados os identificadores dos estados parciais dos dois corpos das duas instâncias de módulo do **SP** que participam da comparação com o **SE**. Inicialmente, declaram-se os estados do corpo local e posteriormente os do corpo remoto, utilizando-se a sintaxe mostrada acima, inclusive com as linhas tracejadas, separando os campos de declaração. Uma das linhas é feita com o sinal de menos "-" e a outra é feita com o sinal de igual "=". A quantidade de caracteres em cada uma das linhas é arbitrária. Caso as duas instâncias de módulo do **SP**, envolvidas na análise, possuam um mesmo corpo, então basta declarar um conjunto de identificadores de estados parciais e, nesse caso, não deve aparecer o número "1" entre parênteses nas palavras-chaves de declaração desses estados.

O último campo de declaração do arquivo ***.SLN*** trata-se da relação nebulosa $\tilde{R}_{EX}(e,x)$. O usuário deve escrever o valor de pertinência para cada par de estados parciais do **SE** e do **SP**, ou seja, para cada estado parcial **e** do **SE**, deve ser montado um par com cada estado parcial **x** do **SP**, fornecendo, em seguida o valor μ do par (e,x). O valor μ caracteriza o grau de semelhança entre os estados parciais **e** do **SE** e **x** do **SP**. O usuário não precisa escrever os pares de estados parciais que não possuem qualquer grau de semelhança, isso é, aqueles pares (e,x) com pertinência igual a zero.

Arquivo HMS.SEQ

*Estado inicial = (1) -{1}T1S-> (2) -{3}M1-> (3) -{2}T1R-> (4) -{3}M2-> (5) -{1}T2S-> (6) -{3}M3-> (7) -{2}T2R-> (8) ***** OBJETIVO ALCANCADO *****

No arquivo de seqüência de disparos, as transições são registradas com o seu nome e o valor numérico da referência da instância de módulo que contém a transição entre chaves. Um número, entre parênteses, que indica qual estado global foi obtido após o disparo da transição, também é mostrado nesse arquivo, conforme pode ser visto acima.

4.2 Protocolo ABRACADABRA

O protocolo ABRACADABRA [14] foi padronizado em conjunto pela ISO e pelo antigo CCITT para ser usado exclusivamente em testes. O ABRACADABRA é 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 figura 4 a seguir mostra a arquitetura dessa especificação:

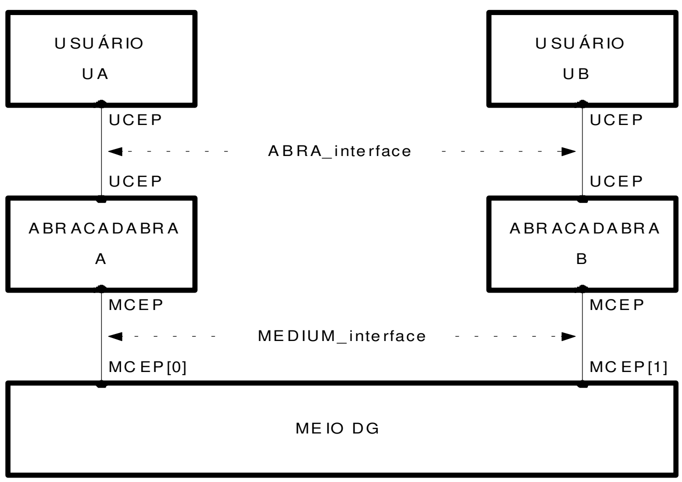


Figura 4: Arquitetura da especificação do protocolo ABRACADABRA.

Nessa especificação, o corpo da entidade usuário foi implementado de forma simplificada, tendo apenas o objetivo de fornecer à camada do protocolo ABRACADABRA, as primitivas de requisição e resposta de serviço para promover o seu teste.

O corpo da entidade usuário possui os seguintes estados:

- ***REPOUSO*** — estado inicial em que o usuário poderá enviar o pedido de abertura de conexão ou receber a indicação de um pedido de abertura de conexão transmitido pela entidade usuário distante;
- ***CONFIRMA_CONEXAO*** — estado em que o usuário emite a confirmação de um pedido remoto de conexão;

- ***ESPERANDO_CONEXAO*** — estado em que o usuário emissor do pedido de conexão, fica aguardando a chegada da confirmação da entidade remota;
- ***TRANSMITINDO_DADO*** — estado em que a entidade usuária solicitante da abertura de conexão, faz a emissão de um dado para, logo em seguida, enviar um pedido de desconexão.

Com relação aos estados das entidades do ABRACADABRA, 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.

As verificações realizadas 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 **A** e **B** no estado parcial ***OPEN***. O estado global inicial teria as instâncias **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 **A** e **B** no estado ***OPEN***, sendo que **A** teria no ponto **UCEP** a primitiva ***DatReq*** (enviada por **UA**, solicitando envio de dado) e ***S3*** foi declarado com o módulo **A** no estado ***WFAK*** e o módulo **B** no estado ***OPEN*** com a primitiva ***UnitInd*** (enviada pelo **meio DG**) no ponto **MCEP**.

A verificação nº (1) foi feita de duas maneiras: a primeira sem que o **Verest+** acionasse os sistemas de lógica nebulosa e a segunda, chamando-os normalmente. No primeiro caso, como mostra o trecho final do arquivo de descrição de resultados, apresentado a seguir (ABRAR11.DES), 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 ABRAR12.DES também a seguir.

Trecho do arquivo ABRAR11.DES

```
============================================================
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 ABRAR12.DES

===
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 ABRAR2.DES, 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 [1] usada pelo **Verest+** só manipula 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 ABRAR2.DES

===
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 CONCLUSÕES

Este artigo mostrou a implementação e a utilização de um verificador (**Verest+**) feito para testar especificações de protocolos descritos na linguagem ESTELLE, comprovando assim, a viabilidade do emprego da técnica de verificação apresentada em [1], pois todos os testes feitos forneceram a resposta esperada. Todas as propriedades de interesse checadas eram verdadeiras e, realmente, deveriam ser uma vez que tais protocolos [13,14] já haviam sido testados antes com tais serviços já provados.

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.

A partir dos testes realizados, pode-se concluir que a técnica de verificação [1] implementada no **Verest+** é promissora, indicando que seria compensador o aperfeiçoamento da mesma, definindo novas regras de inferência para manipular os outros objetos presentes na especificação dos protocolos, como primitivas de serviço e algumas estruturas de dados.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] BERNARDO FILHO, O., PEDROZA, A. C. P., LEÃO, J. L. S. e FERNANDEZ, M. P., *Uma Técnica com Lógica Nebulosa para Verificação de um Protocolo de Gerência Hierárquica de Rede*, XVI Simpósio Brasileiro de Redes de Computadores, 1998.

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

[3] 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.

[4] RESCHER, N. e URQUHART, A., *Temporal Logic*, Springer-Verlag Wien New York, 1971.

[5] MANNA, Z. e PNUELI, A., *Verification of Concurrent Programs: Temporal Proof Principles*, Proceedings of Workshop on Logics of Programs, Lectures Notes in Computer Science, Springer-Verlag, 1981.

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

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

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

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

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

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

[12] OLIVEIRA JR., R. C., *Um Compilador para a linguagem ESTELLE*, Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1991.

[13] 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.

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