Plataforma de Software para Desenvolvimento de Algoritmos de Busca para ESTELLE

Emmanuel Gomes Sanches

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO,

Aprovada por :

Aloysio de Castro Pinto Pedroza, D.Sc.
(presidente)

Edmundo A. de Souza e Silva, Ph.D.

Luís Felipe Magalhães de Moraes, Ph.D.

Rosa Maria Leão Rust Carmo, D.Sc.

Rio de Janeiro, RJ - Brasil

Abril de 1995

SANCHES, EMMANUEL GOMES SANCHES

Plataforma de Software para Desenvolvimento de Algoritmos de Busca para ESTELLE [Rio de Janeiro] 1995

XV, 171p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1995)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1- Redes de Computadores 2- Sistemas Distribuídos

3- Protocolos de Comunicação

I. COPPE/UFRJ II. Título (Série).

Dedicatória

aos meus pais

Agradecimentos

Ao prof. Aloysio de Castro pela organização e dedicação durante toda a fase de desenvolvimento e por todo o apoio, incentivo e esforços determinantes em todos os momentos.

Ao prof. Edmundo de Souza por todos os esclarecimentos, sugestões e contribuições importantes.

Ao prof. Orlando Bernardo pela amizade, explicações e auxílios decisivos sobre o Verificador, uma das referências na qual se baseia o trabalho desenvolvido.

Especialmente à minha mãe Celina e ao meu pai José pelo eterno auxílio e por terem me proporcionado esta oportunidade.

E à instituição CAPES pelo apoio financeiro.

Sem a inestimável contribuição de qualquer um destes acima mencionados não seria possível realizar todo este trabalho. Obrigado.

RESUMO DA TESE APRESENTADA À COPPE/UFRJ COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M. Sc.).

Plataforma de Software para Desenvolvimento de Algoritmos de Busca para ESTELLE

Emmanuel Gomes Sanches

Abril de 1995

ORIENTADORES :

Aloysio de Castro Pinto Pedroza

Edmundo A. de Souza e Silva

PROGRAMA :

Engenharia de Sistemas e Computação

RESUMO : Este trabalho mostra o desenvolvimento de uma Plataforma de Software que tem por objetivo permitir a implementação de Algoritmos de Busca aplicados a Protocolos de Comunicação especificados com o uso da Técnica de Descrição Formal Estelle. Com esta plataforma a criação de ferramentas para o estudo e desenvolvimento de protocolos de comunicação fica facilitada, uma vez que são oferecidas estruturas de dados internas correspondentes à especificação Estelle. Para que tal plataforma possa receber algoritmos de interesse também da área de desempenho, a Técnica Estelle foi acrescida da capacidade de descrever características dos sistemas como a probabilidade de disparo de cada transição da especificação Estelle..

ABSTRACT OF THESIS PRESENTED TO COPPE/UFRJ AS PARTIAL FULFILLMENT OF REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (M. Sc.).

Software Platform to Development of Search Algorithms to ESTELLE

Emmanuel Gomes Sanches

April de 1995

THESIS SUPERVISORS :

Aloysio de Castro Pinto Pedroza

Edmundo A. de Souza e Silva

DEPARTMENT :

Systems Engineering and Computer Science

ABSTRACT : This work presents the developing of a software platform which has the main objective to help the implementations of search algorithms with applications in communication protocols specified in the Formal Description Technique Estelle. Using this platform, the creation of tools to study and help the development of communications protocols are facilitated with the offering of special internal data structures. For the platform to receive algorithms of interest including perform analisys, the technique Estelle was incresed with the capability of describe system caracteristics like probability of each transition of specification.

Índices

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1

1. Introdução

Uma tarefa muito trabalhosa, que ainda se costuma fazer de maneira braçal e muito sujeita a falhas humanas, é a de projetar e, posteriormente, depurar um protocolo de comunicação assim como a análise de seus estados. Entretanto, com o surgimento das Técnicas de Descrição Formal vislumbrou-se um novo caminho para realizar tal tarefa.

Associando a padronização proporcionada por estas técnicas com a teoria de redes, é possível fazer a simulação, verificação, avaliação de desempenho e outras análises sobre estes sistemas especificados de maneira formal.

1.1. Motivação

A COPPE Elétrica está desenvolvendo um pacote de ferramentas com o intuito de auxiliar na concepção e depuração de projetos de protocolos de comunicação.

Para abordar todos os projetos de protocolos de uma maneira única e padronizada, escolheu-se a Técnica de Descrição Formal chamada Estelle [03], [04], [05], [06], [07]. Portanto os protocolos que se deseja estudar deverão estar descritos conforme esta técnica para que então possam ser utilizadas as ferramentas desenvolvidas.

Algumas das ferramentas [15], [13], [16], [12], [20] e [21] do pacote [11] já foram desenvolvidas e outras não. Das ferramentas que já foram desenvolvidas, algumas foram implementadas somente para PC em ambiente DOS e outras já possuem suas respectivas versões para rodarem em estações de trabalho SUN sob sistema operacional UNIX.

1.2. Objetivo

O objetivo deste trabalho é disponibilizar uma Plataforma de Software para auxiliar o desenvolvimento de ferramentas que se baseiam em algoritmos de busca em sistemas de comunicação descritos formalmente segundo a técnica Estelle.

Tais ferramentas se destinarão a serem incluídas no Sistema de Auxílio a Projetos de Protocolos de Comunicação citado [11].

Outra meta é dotar o ambiente que está sendo desenvolvido da capacidade de especificar informações de interesse da área de análise de desempenho, para que possam ser desenvolvidas ferramentas também com este propósito.

Para exemplificar a confecção de uma ferramenta utilizando a Plataforma de Software em questão, foram implementados dois algoritmos que fazem a geração de árvores de alcançabilidade de protocolos de comunicação descritos em Estelle.

Inicialmente foi implementada uma versão para ser operada em PCs para disponibilizar a utilização da Plataforma de Software ao Sistema de Auxílio que possui um maior número de ferramentas já implementadas para a base do DOS. Nesta versão, as ferramentas desenvolvidas são úteis para o ensino de graduação que, com posse de um pacote de ferramentas desta natureza, já poderá fazer estudos de protocolos de comunicação de pequenos sistemas, utilizando-se de uma plataforma de baixo custo.

Em seguida foi feito o transporte da Plataforma de Software, dos computadores IBM PC para as estações UNIX da SUN. Esta nova versão está mais capacitada a auxiliar no desenvolvimento de ferramentas de projeto e análise de protocolos de comunicação para sistemas reais com um maior número de estados e instâncias ou nós, neste ambiente computacional.

1.3. Organização do Trabalho

Este documento foi concebido de maneira que, inicialmente, sejam expostos os motivos que fizeram com que este trabalho fosse realizado e quais metas deveriam ser alcançadas no término deste.

No capítulo dois são apresentados os conceitos teóricos fundamentais dos quais se precisa ter conhecimento para o entendimento dos procedimentos tomados na confecção do trabalho, tais como: quais e o que são as técnicas de descrição formal, os aspectos da TDF Estelle e como todos estes conceitos foram utilizados.

O terceiro capítulo detalha a estrutura da Plataforma de Software desenvolvida, os recursos que oferece, maneira como foi implementada, arquitetura, organização dos dados de Estelle e ainda são apresentadas as diferenças estruturais entre as versões da mesma para os PCs e SUNs. É apresentado ainda um algoritmo de busca implementado em tal plataforma para exemplificar sua utilidade no desenvolvimento de ferramentas do Sistema de Auxílio.

No quarto capítulo, são descritos os sistemas que foram testados pela ferramenta-exemplo desenvolvida na plataforma, relatados todos os testes realizados e seus respectivos resultados e conclusões.

O capítulo quinto apresenta todas as conclusões, assim como os benefícios e perspectivas que o trabalho deixa.

Na forma de anexos são apresentados um manual de utilização do software para orientar os usuários e as listagens das especificações dos exemplos adotados seguidos das listagens das respectivas saídas oferecidas pela plataforma.

Capítulo 2

2. Conceitos Básicos

2.1. Introdução

O surgimento das TDFs proporcionou uma abordagem mais homogênea e padronizada das questões de projeto e depuração de sistemas distribuídos, em especial para os protocolos de comunicação. Esta questão é abordada adiante na seção 2.2.

Neste trabalho, uma destas TDFs foi eleita para ser utilizada no desenvolvimento de ferramentas com esta finalidade. Para a plena utilização destas ferramentas, todos os protocolos devem ser descritos segundo as normas de **Estelle** para que então possam se submeter aos testes. Os conceitos de Estelle são mostrados em 2.3.

Finalmente na seção 2.4, é explicada toda a **metodologia** utilizada na confecção desta Plataforma de Software para desenvolvimento de ferramentas baseadas em Algoritmos de Busca em sistemas descritos formalmente em Estelle.

2.2. As Técnicas de Descrição Formal

2.2.1. Introdução

Freqüêntemente, quando se deseja fazer a descrição de qualquer sistema de comunicação, seja para apresentação ou simples documentação do mesmo, esta descrição é feita de maneira informal e não padronizada, podendo ficar sujeita a interpretações dúbias que não correspondem à realidade.

Para tentar resolver esta questão começaram a ser feitos estudos no sentido de desenvolver técnicas de descrição formal (TDFs).

Na verdade, a idéia de criar uma linguagem para descrever a arquitetura ou mesmo o funcionamento de um sistema e processar este código gerado surgiu para especificar circuitos de computadores e não para sistemas de comunicação. Estas linguagens foram portanto denominadas por 'Computer Hardware Description Languages' (CHDLs) e os primeiros trabalhos a respeito, tais como [01] e [02], surgiram na década de 60. Estes conceitos foram adaptados para os sitemas de comunicação, até que fossem criadas as TDFs.

2.2.2. Algumas Técnicas

As TDFs deveriam ser um conjunto de regras universais que possibilitassem que sistemas distribuídos pudessem ser descritos de uma maneira padronizada, clara e não redundante, de modo que, quem quer que viesse a ler uma descrição eleita segundo estas técnicas, entendesse facilmente a arquitetura e o funcionamento do sistema descrito.

Vários foram os trabalhos desenvolvidos neste sentido e algumas das TDFs precursoras são citadas a seguir.

* NBS - (National Bureau of Standards)
* FDT B - (subgrupo B - ISO)
* X.250 - (CCITT)

Mas com o tempo, surgiram conceitos mais modernos e estas TDFs foram perdendo o uso.

Atualmente, as mais difundidas são :

* Estelle - Extended State Transition Language

* LOTOS - Language of Temporal Ordering Specification

* SDL - Specification and Description Language

Infelizmente, devido à divergências de opiniões quanto aos conceitos básicos sob os quais deve estar baseada uma TDF, ainda não se chegou a um consenso sobre a TDF que deve se tornar padrão, apesar dos esforços que têm sido feitos neste sentido.

Diferentes entidades e grupos de pesquisa ingressaram no estudo das TDFs. Estelle e LOTOS foram ambas desenvolvidas por subgrupos diferentes da International Organization for Standardization (ISO), e a SDL foi criada pelo Comitè Consultatif International Tèlègraphique et Tèlèphonique (CCITT).

Existem ainda variações. A SDL, por exemplo, possui uma forma gráfica muito difundida e de fácil utilização, mas dependendo do sistema a ser descrito, sua respectiva especificação pode vir a ficar muito grande.

Até cerca de dez anos atrás, os conceitos de Estelle e SDL possuíam muito em comum. Em ambas, só havia um único tipo de módulo (o conceito de módulo será definido posteriormente), uma única forma de paralelismo e só podiam ser criadas instâncias de módulos de maneira estática.

Com o surgimento de novas idéias e a necessidade de descrever sistemas mais complexos, Estelle evoluiu para um novo padrão com o qual se tornou possível : a criação, de forma dinâmica, de instâncias de módulos; a definição de quatro tipos de módulos e a ocorrência de diferentes formas de paralelismo. Com estas modificações dos conceitos, a capacidade de expressão da linguagem ficou bem maior.

2.3. A Linguagem Estelle

2.3.1. Introdução

Estelle [03] é uma linguagem que foi concebida com o intuito de fornecer condições necessárias para se fazer uma descrição formal de qualquer sistema de processamento distribuído ou concorrente de tal forma que se tornou uma ferramenta poderosa para se especificar, de maneira clara e objetiva, os serviços e o funcionamento de um protocolo de comunicação.

Esta TDF desenvolveu-se com base numa máquina de estados finita extendida (MEFE), combinando estados, interações e transições com variáveis, parâmetros e prioridades do Pascal padrão.

Uma especificação feita com Estelle [03] descreve um sistema através de um conjunto de módulos que possuem portas de entrada e saída e se comunicam através de canais bidirecionais, assim como sua evolução com o decorrer do tempo através de transições de estados.

A seguir, serão apresentados os conceitos básicos da linguagem que possibilitam a descrição de um sistema e após isto, será apresentada a estrutura de um programa em Estelle.

2.3.2. Conceitos Básicos de Estelle

Esta linguagem possui um conjunto de objetos que auxiliam na descrição da "disposição física" dos componentes do sistema modelado em máquinas de estado assim como o comportamento interno destas máquinas de estado com o decorrer dos eventos.

A maneira pela qual estas máquinas de estado realizam transferências de

informações entre si será comentada com maiores detalhes posteriormente.

2.3.2.1. Objetos de Descrição da Disposição do Sistema

Para se descrever qualquer sistema com as técnicas de Estelle, é necessário inicialmente conceber um modelo deste sistema constituído de várias máquinas de estado interligadas de maneira conveniente.

Uma vez feito isto é que poderão ser utilizados os elementos da linguagem (representados a seguir) para se descrever de que maneira estão dispostas espacialmente e como estão interligadas as máquinas de estado idealizadas.

a) O Módulo

Um módulo é como uma 'caixa preta' que possui portas de entrada e saída que podem ou não estar conectadas às portas de entrada e saída de outros módulos. Através destas portas, transitam mensagens que são enviadas e recebidas fazendo assim a comunicação do sistema.

Todo módulo é composto por um cabeçalho e um corpo. O cabeçalho nada mais é que uma declaração de um tipo de módulo, trazendo informações tais como quantas portas ele possui, entre outras, podendo haver vários módulos do mesmo tipo.

O corpo de um módulo descreve seu comportamento interno. Se dentro de um módulo residirem submódulos, o corpo do módulo principal trará a declaração de cada um destes submódulos de maneira análoga à declaração do módulo. Da mesma forma que acontece com o módulo ocorre com o corpo dos módulos, podendo existir módulos do mesmo tipo com corpos diferentes. Esta facilidade permite que possam ser descritas instâncias semelhantes com comportamentos

diferentes. Na realidade, quem troca informações são as instâncias de um tipo de módulo.

Estrutura Hierárquica dos Módulos

Esta possibilidade de módulos poderem conter em seu interior submódulos define uma estrutura hierárquica e ainda um grau de parentesco entre os módulos.

Existem duas maneiras de representar a estrutura hierárquica de um sistema: em forma de blocos e em forma de árvore genealógica.

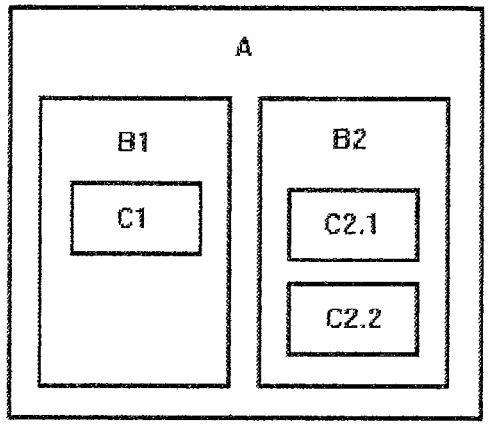

Figura 2.1 - Representação de uma estrutura hierárquica em forma de blocos.

OBS : Os módulos 'B1' e 'B2' possuem o mesmo grau hierárquico e também o mesmo 'módulo pai', portanto podem ser chamados de 'módulos irmãos'. Já os módulos 'C1' e 'C2.1' possuem o mesmo grau hierárquico mas não são submódulos do mesmo módulo, portanto não podem ser considerados como 'módulos irmãos'.

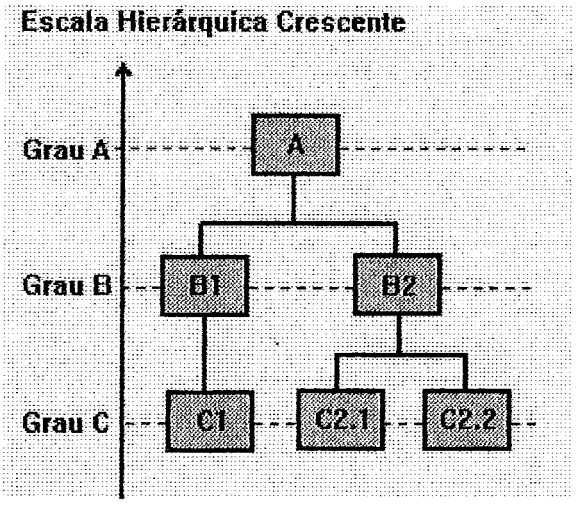

Figura 2.2 - Representação de uma estrutura hierárquica em forma de árvore genealógica.

Atributos de um módulo

Os atributos de um módulo servem para, durante a declaração do tipo de módulo, serem descritas as características de paralelismo de execução entre os módulos, assim como de que maneira estes podem ser refinados (vide figura 2.3).

Características \ Atributos		SYSTEMPROCESS	SYSTEMACTIVITY	PROCESS	ACTIVITY
módulos com este atributo podem ser refinados em submódulos com atributo do tipo :	PROCESS	X		X	
	ACTIVITY	X	X	X	X
módulos com este atributo devem ser submódulos de outro módulo.				X	X

Tabela 2.1 - Características de um módulo devido seu respectivo atributo.

Com o desenvolvimento de uma nova versão da linguagem, o número de possíveis atributos de um módulo aumentou e atualmente são os seguintes : Systemprocess, Systemactivity, Process e Activity.

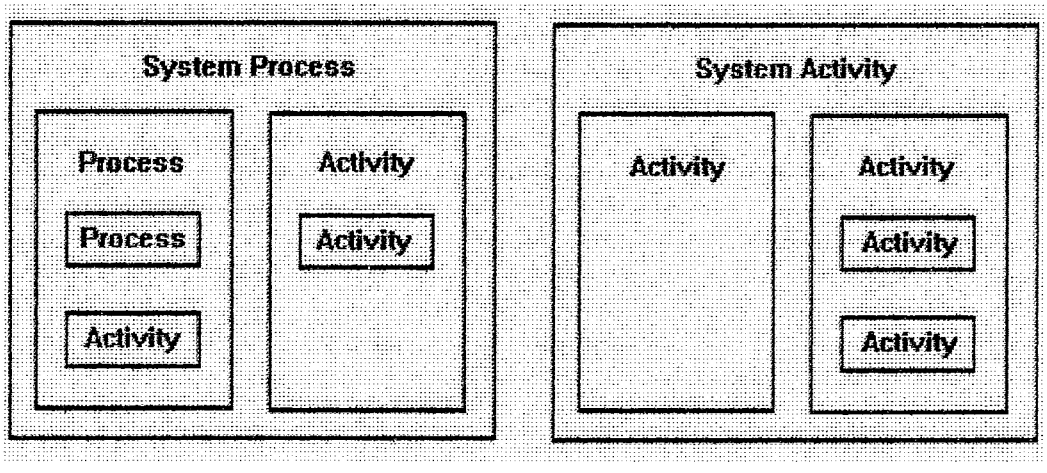

Figura 2.3 - Exemplos de possíveis refinamentos de módulos conforme o atributo.

Classificação dos Módulos

Os módulos podem ser classificados como ativos e inativos.

Módulos Ativos - São aqueles que possuem transições, o que possibilita que eles saiam do estado inicial. Possuem atributos, o que possibilita a existência de 'módulos filhos'.

Módulos Inativos - São aqueles que não possuem transições nem atributos, ficando estes permanentemente no estado inicial.

Paralelismo

Existem três tipos de paralelismo a serem vistos : o paralelismo síncrono, o assíncrono e o não determinístico.

O paralelismo síncrono ocorre quando transições de um módulo são disparadas ao mesmo tempo.

Se as transições de um módulo são disparadas em tempos distintos, ocorre o paralelismo assíncrono e estas evoluem independentemente.

Quando uma única transição é disparada a cada vez, é caracterizado o não determinístico.

O paralelismo entre módulos ocorre de maneira assíncrona quando estes possuem os atributos Systemprocess e Systemactivity.

Se 'módulos filhos' (de mesmo grau hierárquico) estiverem contidos num mesmo 'módulo pai', o paralelismo entre eles ocorrerá de maneira síncrona se o 'módulo pai' for do tipo Systemprocess.

Já se for do tipo Systemactivity, não se verifica a ocorrência de qualquer forma de paralelismo entre os 'módulos filhos'.

Sintaxe do Módulo

Como foi visto anteriormente, um módulo é composto de um cabeçalho e de um corpo. Portanto a sintaxe de cada um será vista separadamente.

Sintaxe do Cabeçalho - O cabeçalho detalha quantas portas de entrada e saída possui um módulo, atribui nomes a estas portas, especifica por qual tipo de canal cada porta pode se comunicar a portas de outro módulo, a que extremidade do canal cada porta está ligada e a que tipo de fila interna estas portas estão

conectadas. O cabeçalho deve ser declarado assim :

```
MODULE mod_type ( lista de parâmetros ) atributo :
        IP porta_1 : can_type ( extrem_1 ) fila_type ;
        IP porta_2 : can_type ( extrem_2 ) fila_type ;
END ;
```

Sintaxe do Corpo - O corpo de um módulo nos diz quais são as condições iniciais do módulo em questão, quais os possíveis estados que este módulo pode assumir e sob que condições esta mudança de estado acontece, descrevendo portanto de maneira completa, a evolução deste módulo através de transições de estado.

Um corpo de módulo é declarado de forma dedicada, sendo cada um específico para um tipo de módulo, ao contrário do tipo de módulo que pode possuir mais de um possível corpo. Agora veremos como declarar um corpo de módulo :

```
BODY corpo_type FOR mod_type ;
BEGIN
        .
        .       ( definição do corpo )
        .
END ;
```

Ainda no corpo são feitas as declarações de 'módulos filhos', se este for o caso.

b) O Ponto de Interação

O ponto de interação ou ainda IP (Interaction Point) é a entidade que representa uma porta de entrada e saída dentro de uma especificação Estelle. Ele é definido durante a declaração de módulo e as primitivas que dele saem são enumeradas durante a declaração de tipo de canal, dependendo a qual extremidade ele está ligado. Naturalmente as primitivas que estão sendo esperadas são aquelas descritas na outra extremidade do canal, a qual outro módulo poderá estar conectado.

Um ponto de interação pode estar posicionado na periferia do módulo (quando está conectado a outro módulo) .

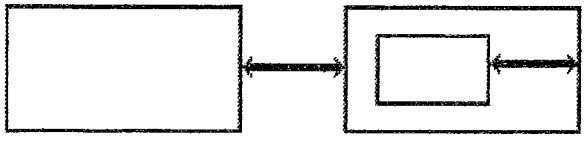

Figura 2.4 - Exemplo de módulos com pontos de interação posicionados na periferia do módulo.

ou ainda estar posicionado completamente no interior do módulo.

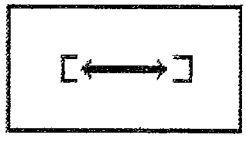

Figura 2.5 - Exemplo de módulo com pontos de interação posicionados no interior do módulo.

c) O Canal

É a entidade que interliga as máquinas de estado. Possibilita que primitivas de comunicação pertinentes àquela ligação trafeguem por ele em ambos os sentidos (propriedade de bidirecionalidade) e enumera quais podem vir por um lado e quais podem vir pelo outro lado.

Sintaxe do Canal

Na verdade não se declara um canal. Este fica implícito na especificação dentro da declaração de tipo de módulo quando associamos a um ponto de interação uma extremidade de um tipo de canal.

O que se declara é o tipo do canal, ou seja, quais primitivas trafegam por ele e de qual lado elas podem vir. A declaração de tipo de um canal é feita da seguinte maneira :

```
CHANNEL can_type ( extrem_1, extrem_2 ) ;
BY extrem_1 : primit_1A ( lista de parâmetros ) ;
                    primit_1B ( lista de parâmetros ) ;
                        . . .
BY extrem_2 : primit_2A ( lista de parâmetros ) ;
                    primit_2B ( lista de parâmetros ) ;
                        . . .
```

Isto significa que um canal declarado como sendo do tipo 'can_type', possibilita que do módulo conectado à extremidade 'extrem_1' venham as primitivas de comunicação 'primit_1A', 'primit_1B', ... e do módulo conectado à extremidade

'extrem_2' venham as primitivas de comunicação 'primit_2A', 'primit_2B', ...

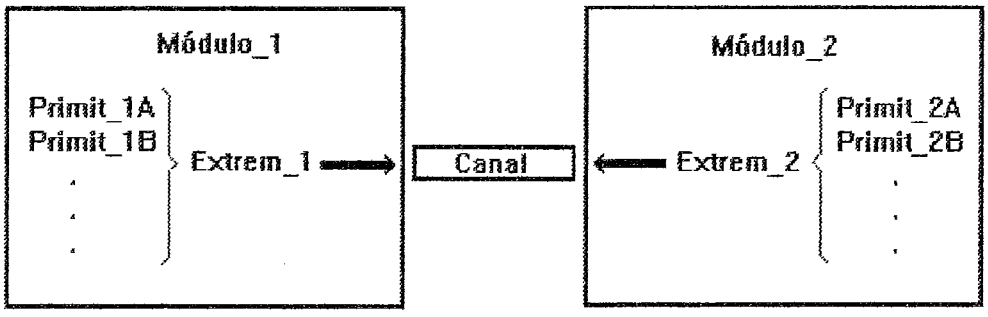

Figura 2.6 - Esquema representativo da interconexão de dois módulos através de um canal.

2.3.2.2. Objetos de Descrição da Evolução do Sistema

Apesar de já citados, serão vistos agora, nesta seção, com maiores detalhes, os objetos da linguagem que auxiliam na realização da descrição do comportamento interno de cada módulo. Esta descrição é feita através da inicialização de cada corpo e de suas respectivas transições.

a) Inicialização

Na inicialização de um sistema, podem ser feitas atribuições às variáveis, atribuições de corpos à instâncias de módulo e interconexões de IPs.

Se, ao invés de ser do sistema, a inicialização for somente de um corpo de módulo, a única diferença será que as interconexões feitas só irão fazer referência ao módulo em questão e seus submódulos.

Sintaxe das Inicializações

A inicialização, tanto de um sistema quanto de um corpo de módulo, é feita como segue :

```
INITIALIZE TO estado_A
BEGIN
    .
    .       ( inicializações )
    .
END ;
```

b) As Transições

As transições possibilitam a especificação detalhada de sob quais condições uma instância muda de estado e quais procedimentos são realizados após esta mudança.

Sintaxe das Transições

Uma sequência de transições deve vir após um 'TRANS'. Para indicar de qual e para qual estado vai um módulo, utiliza-se o 'FROM TO'. As cláusulas que testam se aquela transição será disparada ou não são 'WHEN' e 'PROVIDED'. A lista de sentenças que devem ocorrer no caso da transição ser disparada deve ser precedida de 'BEGIN' e concluida com um 'END ;'.

As transições podem ser basicamente de quatro tipos que serão apresentados nos exemplos que seguem.

Exemplo 1 : Neste exemplo a lista de instruções só será executada após a chegada da primitiva de comunicação 'primit_A' e caso o resultado do teste de uma expressão lógica seja 'TRUE'.

```
TRANS
        FROM estado_inicial TO estado_final
        WHEN porta_A.primit_A
        PROVIDED ( teste lógico )
        BEGIN
                .
                .       ( lista de instruções )
                .
        END ;
```

Exemplo 2 : Já neste outro a lista de comandos depende somente da chegada da primitiva de comunicação 'primit_A' que está sendo esperada.

```
TRANS
        FROM est_inic TO est_fim
        WHEN porta_A.primit_A
        BEGIN
                .
                .       ( lista de instruções )
                .
        END ;
```

Exemplo 3 : Desta vez, bastando ter passado no teste lógico, a lista de instruções será executada.

```
TRANS
        FROM est_inic TO est_fim
        PROVIDED ( teste lógico )
        BEGIN
                .
                .       ( lista de instruções )
                .
        END ;
```

Exemplo 4 : Este exemplo dispara a execução das instruções direto sem fazer qualquer teste antes.

```
TRANS
        FROM est_inic TO est_fim
        BEGIN
                .
                .       ( lista de instruções )
                .
        END ;
```

2.3.2.3. A Comunicação

A comunicação entre os módulos é feita através do envio e recepção de primitivas de comunicação que são definidas durante a declaração de tipo de canal,

primitivas estas que entram e saem dos módulos através de seus pontos de interação, transitam em ambos os sentidos pelos canais e são armazenadas em filas infinitas que são internas aos módulos.

a) As Filas

As filas de um módulo são ilimitadas (limitadas somente pela memória disponível do equipamento utilizado), são do tipo FIFO (First In First Out) e sempre conectadas de alguma maneira, aos pontos de interação. São classificadas em dois tipos (common e individual), de acordo com a forma como os pontos de interação estão ligados a ela.

Individual Queue

São aquelas que armazenam as primitivas de comunicação que chegam por um único ponto de interação.

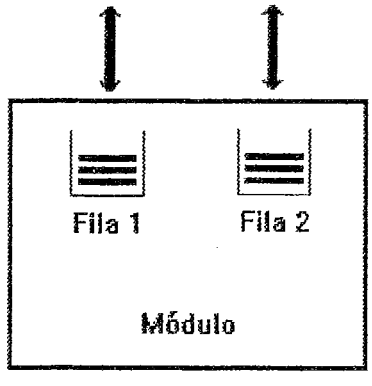

Figura 2.7 - Esquema representativo de duas filas individuais.

Common Queue

São aquelas que podem armazenar primitivas de comunicação advindas de mais de um ponto de interação.

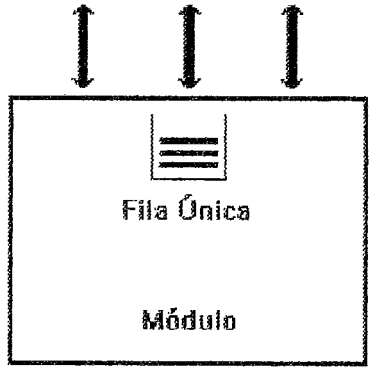

Figura 2.8 - Esquema representativo de uma fila comum.

b) A Troca de Mensagens

O envio e recepção de mensagens constituem o diálogo entre as instâncias de módulo e é realizado com o auxílio de basicamente dois comandos.

WHEN - É uma cláusula que testa se por determinado ponto de interação chegou alguma primitiva de comunicação.

Exemplo : WHEN porta_A.primit_A

Esta sentença verifica se pelo IP 'porta_A' chegou a primitiva de comunicação 'primit_A'.

Se a primitiva em questão possuir parâmetros, não existe a necessidade destes serem mencionados no teste, pois apenas com a verificação de se a primitiva chegou, já se sabe se esta possui ou não parâmetros, e caso existam, estes estarão sendo esperados, podendo ser utilizados, se necessário, durante a lista de instruções da transição, fazendo-se referência aos mesmos nomes dados a eles durante a declaração de tipo de canal.

OUTPUT - Este comando coloca uma primitiva de comunicação na fila FIFO pertencente ao módulo que está esperando a chegada desta.

Exemplo : OUTPUT porta_A.primit_A (parâmetros) ;

Com esta sentença a primitiva de comunicação 'primit_A' é exportada através do IP 'porta_A'.

Desta vez, ao contrário da cláusula WHEN, no OUTPUT é necessário que sejam explicitados os parâmetros, se estes existirem.

Exemplo : .

.

.

```
CHANNEL can_type ( lado_1, lado_2 )
BY lado_1 : primit_A ( param_1, param_2 : param_type) ;
BY lado_2 : primit_B ( param_3, param_4 : param_type) ;

MODULE mod_type SYSTEMPROCESS :
        IP porta_1 can_type ( lado_1 ) ;
END ;
```

```
BODY corpo_type FOR mod_type ;
VAR p1, p2 : param_type ;
        .
        .
        .

TRANS
        FROM estado_1 TO estado_2
        WHEN porta_.primit_B
        BEGIN
                p1 := param_3 + param_4 ;
                p2 := param_3 - param_4 ;
                OUTPUT porta_1.primit_A ( p1, p2 ) ;
        END ;

END ; (* fim do corpo *)
                .
                .
                .
```

c) A Sintaxe das Interligações

As interligações entre as instâncias de módulo são feitas de maneiras diferentes, dependendo do grau hierárquico entre os módulos, e possuem a característica de poderem ser feitas dinamicamente ao longo da execução.

Quando as instâncias de módulo possuem mesmo grau hierárquico, os comandos referentes às interligações são :

CONNECT - Este comando é utilizado para conectar os pontos de interação das instâncias de 'módulos irmãos' (possuem o mesmo 'módulo pai' e mesmo grau hierárquico).

DISCONNECT - Este comando desestabelece a ligação já existente entre 'módulos irmãos'.

Os módulos 'A' e 'B' são de mesmo nível hierárquico e estão dentro de um mesmo 'módulo pai'.

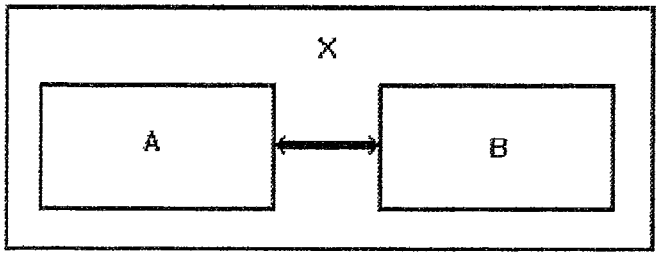

Figura 2.9 - Esquema representativo do tipo de ligação feita pelo comando CONNECT.

Quando entre as instâncias de módulo existe um parentesco equivalente ao de pai e filho, os comandos referentes às interligações são :

ATTACH - Utiliza-se esta cláusula para unir através de um canal os pontos de interação da instância de módulo em questão.

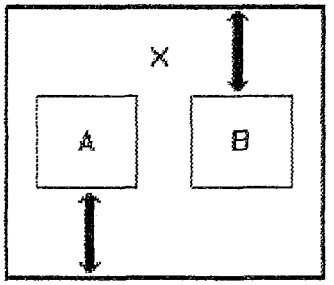

Figura 2.10 - Esquema representativo do tipo de ligação feita pelo comando ATTACH.

Na figura acima, 'X' é 'módulo pai' dos módulos 'A' e 'B'.

DETACH - Esta cláusula desfaz as ligações entre IPs de instâncias de módulo feitas a partir do comando ATTACH.

Todos estes comandos devem ser dados de dentro do corpo do módulo pai referente aos submódulos em questão.

d) Criação e Destruição de Instâncias

A declaração de uma instância de módulo utiliza a cláusula 'MODVAR' e deve ser feita da seguinte maneira :

MODVAR inst_A : mod_type ;

Deste modo está sendo definida a instância 'inst_A' como sendo do tipo 'mod_type'.

A destruição de uma instância de módulo utiliza a cláusula 'RELEASE' e só pode ser feita de dentro de um corpo de um módulo e se referenciar aos

submódulos pertencentes a este módulo.

Uma vez de posse dos conceitos destes objetos, pode-se idealizar modelos de sistemas baseados em máquinas de estado finitas para então poder ser feita a descrição, de maneira objetiva e concisa, das características e do funcionamento deste sistema.

2.3.3. Uma Especificação em Estelle

Agora será feita uma breve abordagem a respeito de como modelar um sistema, estruturando-o em máquinas de estado, e como descrever este modelo.

2.3.3.1. Idealização do modelo

Deve-se procurar identificar, no sistema o qual deseja-se especificar, processos que se comuniquem e/ou possuam alguma relação de paralelismo entre si, quanto à execução de seus procedimentos.

Uma vez feito isto, estes procedimentos devem ser declarados como módulos, que por sua vez interligados com canais de forma análoga a qual os procedimentos se comunicam e, utilizando as demais cláusulas vistas anteriormente, descrever com transições de estado o funcionamento interno de cada módulo.

2.3.3.2. Palavras Reservadas

Como já foi dito, Estelle é como um Pascal padrão com uma extensão composta por várias cláusulas específicas da linguagem. Algumas das cláusulas da linguagem Estelle são enumeradas aqui .

ALL	EXIST	LOWEST	PROVIDED
ATTACH	FORONE	NONE	PURE
CONNECT	FROM	OUTPUT	RELEASE
DETACH	WHEN	PRIMITIVE	SAME
DISCONNECT	INIT	PRIORITY	TO

2.3.3.3. Organização do Programa

Vamos supor que tenha sido concebida esta estrutura como modelo de um sistema para agora ser descrita segundo as técnicas da linguagem Estelle.

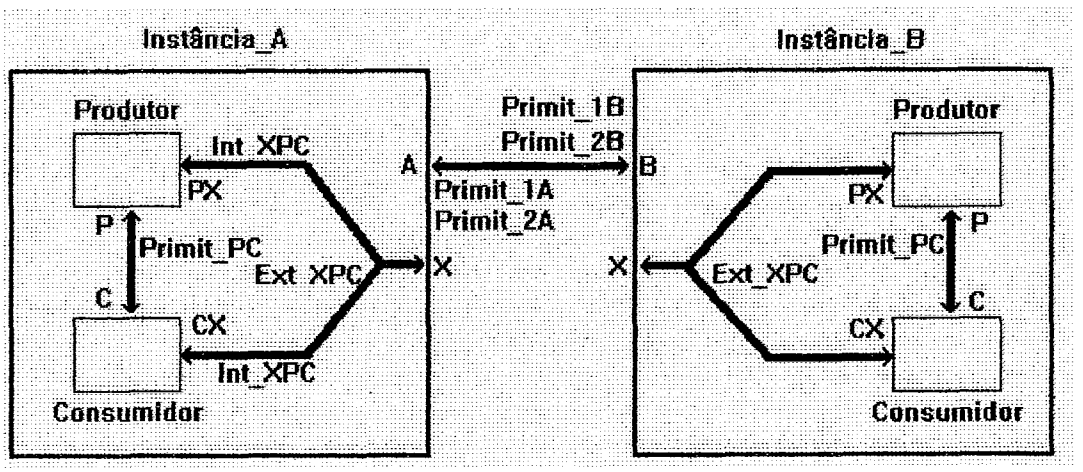

Figura 2.11 - Estrutura hierárquica do sistema exemplo.

A estrutura do programa ficaria da seguinte forma :

```
SPECIFICATION sistema SYSTEMPROCESS

CONST
        .
        .       (* declaração de contantes *)
        .

TYPE
        .
        .       (* declaração de tipos *)
        .

CHANNEL Tipo_Can_AB ( Extrem_A, Extrem_B );
        BY Extrem_A : Primit_1A;
                      Primit_2A;
        BY Extrem_B : Primit_1B;
                      Primit_2B;

CHANNEL Tipo_Can_XPC ( Interno, Externo );
  BY Interno : Int_XPC;
  BY Externo : Ext_XPC;

MODULE Tipo_Mod_A PROCESS;
        IP A : Tipo_Can_AB ( Extrem_A ) INDIVIDUAL QUEUE;
           X : Tipo_Can_XPC ( Externo ) COMMON QUEUE;
END ;
```

```
MODULE Tipo_Mod_B PROCESS;
        IP B : Tipo_Can ( Extrem_B ) INDIVIDUAL QUEUE;
          X : Tipo_Can_XPC ( Externo ) COMMON QUEUE;
END ;

BODY Corpo_Mod FOR Tipo_Mod_A
        CONST
                .
                .       (* declaração de contantes *)
                .
        TYPE
                .
                .       (* declaração de tipos *)
                .

        CHANNEL Tipo_Can_PC ( Produzir, Consumir );
          BY Produzir : Primit_PC;

        MODULE Tipo_Mod_Prod PROCESS;
          IP P : Tipo_Can_PC (Produzir) INDIVIDUAL QUEUE;
                PX: Tipo_Can_XPC (Interno) INDIVIDUAL QUEUE;
        END ;

        MODULE Tipo_Mod_Cons ACTIVITY;
          IP C : Tipo_Can_PC (Consumir) INDIVIDUAL QUEUE;
                CX: Tipo_Can_XPC (Interno) INDIVIDUAL QUEUE;
        END ;
```

```
        BODY Corpo_Prod FOR Tipo_Prod;
        EXTERNAL;

        BODY Corpo_Cons FOR Tipo_Cons;
        EXTERNAL;

        MODVAR Produtor   : Tipo_Mod_Prod;
               Consumidor : Tipo_Mod_Cons;

        INITIALIZE
          BEGIN
              INIT Produtor   WITH Corpo_Prod;
              INIT Consumidor WITH Corpo_Cons;
              ATTACH X TO Produtor.PX;
              ATTACH X TO Consumidor.CX;
              CONNECT Produtor.P TO Consumidor.C;
                    .
                    .
                    .
          END; (* fim da inicialização *)
END ; (* fim do corpo *)

BODY Corpo_Mod FOR Tipo_Mod_B
        .
        . (* Idem "BODY Corpo_Mod FOR Tipo_Mod_A" *)
        .
END ; (* fim do corpo *)
```

```
MODVAR Instancia_A : Tipo_Mod_A;
       Instancia_B : Tipo_Mod_A;

INITIALIZE
  BEGIN
    Instancia_A WITH Corpo_Mod;
    Instancia_B WITH Corpo_Mod;
    CONNECT Instancia_A.A TO Instancia_B.B;
  END; (* fim da inicialização *)

END . (* fim da especificação *)
```

2.4. Metodologia Empregada

2.4.1. Introdução

Nos próximos tópicos da seção 2.4 serão expostas e justificadas todas as escolhas feitas dos procedimentos tomados para confeccionar a Plataforma de Software desenvolvida, tais como : porque adotar Estelle, porque utilizar MODULA-2, qual a estratégia de implementação, quais os acréscimos feitos em Estelle e que considerações foram tomadas antes do desenvolvimento das ferramentas que exemplificam a utilização da plataforma computacional.

2.4.2. A Escolha da TDF

Uma vez que as ferramentas desenvolvidas com a Plataforma de Software se destinam a posteriormente compor o Sistema de Auxílio ao Projeto de Protocolos o qual já elegeu a TDF Estelle como padrão de entrada de tais ferramentas, a questão da escolha de qual Técnica de Descrição Formal (TDF) utilizar como padrão para a entrada da Plataforma de Software se tornou desnecessária.

Portanto utilizar Estelle como padrão de descrição dos protocolos a serem especificados, que servirão de entrada das ferramentas, deixou de ser uma opção para passar a ser uma das metas.

Mas em algum momento inicial do projeto do Sistema de Auxílio foi tomada esta decisão e foram levados em conta alguns aspectos.

Um deles foi quanto à facilidade de descrição dos protocolos pertencentes às camadas do modelo de referência para interconexão de sistemas abertos (RM-OSI) da ISO.

Outro aspecto levado em consideração foi o de optar por uma TDF capaz de especificar sistemas concorrentes de uma forma geral.

E ainda por ter se tornado o padrão mais difundido na área de redes de computadores.

Basicamente por estes motivos foi eleita a TDF Estelle, descrita em [04], [05], [06] e [07] que atende a todas as características desejadas, acima mencionadas, de maneira direta e prática.

Baseados nestes mesmos argumentos, outros trabalhos subseqüêntes têm sido desenvolvidos para Estelle como [13], [14], [15], [16] e [20] e da mesma forma foi com a Plataforma de Software para Algoritmos de Busca.

2.4.3. A Escolha da Linguagem de Programação

Da mesma forma que ocorreu com a opção por Estelle como linguagem de descrição formal padrão, aconteceu com a escolha da linguagem computacional a ser usada.

Como todo o software das ferramentas que compõem o Sistema de Auxílio ao Projeto de Protocolos já havia sido desenvolvido com base na linguagem MODULA-2, descrita em [08] e [09], optou-se por adotar a mesma como linguagem de programação na confecção de uma plataforma computacional que irá, além de auxiliar no desenvolvimento de futuras ferramentas do Sistema de Auxílio, possibilitar uma integração otimizada das ferramentas já implementadas.

As ferramentas que compõem o Sistema de Auxílio se utilizam das mesmas informações disponibilizadas pela Forma Intermediária que é a saída do Compilador Estelle [14].

Este compilador foi desenvolvido em MODULA-2 e, para facilitar a compatibilidade da saída do mesmo com a entrada de dados oferrecida pela

plataforma às novas ferramentas, tal plataforma foi desenvolvida em MODULA-2 também.

A idéia de desenvolver o CAD de Estelle em MODULA-2 surgiu, inicialmente, baseada nas questões que seguem.

A linguagem escolhida deveria ser uma linguagem de alto nível que facilitasse o desenvolvimento de sistemas grandes, ou seja, seria interessante que a linguagem escolhida já possuísse, ela própria, meios para fazer a modularização de grandes sistemas. Então MODULA-2 e PASCAL eram bons candidatos.

A linguagem não poderia ser muito diferente para cada sistema, isto é, ela deveria ser "portável" (fácil de ser transferida de um sistema para outro). Neste caso PASCAL não era indicada pois existem diferentes versões de um sistema para outro e até mesmo de um fabricante de compilador para outro. Logo só sobrou MODULA-2 como linguagem imperativa de alto nível estruturada com facilidades para fazer modularização de grandes sistemas e com uma razoável portabilidade.

No início, MODULA-2 parecia ser então uma boa idéia. Entretanto, a experiência acabou mostrando que a linguagem C é que era verdadeiramente portável após chegarem as estações SUN e, além disso, o ambiente de desenvolvimento disponível para MODULA-2 é pouco moderno.

Acredito que se fosse escolhida novamente, hoje a linguagem C seria a vencedora da concorrência pois, apesar de não possuir as mesmas facilidades para modularizar sistemas grandes e também ser uma linguagem de médio nível (o programador tem que se preocupar com mais detalhes físicos), ela possui melhores ambientes de desenvolvimento e é mais portável.

2.4.4. Estratégia de Desenvolvimento

O compilador Estelle [14], que permite que seja validada a sintaxe da

descrição dos protocolos especificados por esta TDF, já estava com uma nova versão disponível. Este compilador gera como saída, uma Forma Intermediária da descrição, que precisa ser devidamente interpretada para que seja feita a montagem da base de dados a respeito do protocolo em teste, para que só então esta base possa ser devidamente trabalhada por algoritmos de busca com características especiais, que junto com os serviços oferecidos pela plataforma de desenvolvimento resultará numa nova ferramenta integrada ao Sistema de Auxílio ao Projeto de Protocolos.

Todas estas etapas são muito extensas e complexas, sendo que, de um modo geral, cada uma delas dá argumento suficiente para a confecção de uma tese, tamanho seus respectivos graus de dificuldade.

Em [17] foi desenvolvida uma metodologia orientada a objeto que corresponde à etapa de geração da forma intermediária da descrição, acima citada, onde o modelo do protocolo é descrito em termos de objetos que interagem entre si, através de troca de mensagens.

Em [18] é feita a geração de Cadeias de Markov, baseada na metodologia orientada a objeto anteriormente descrita.

Posteriormente, com o trabalho de [14], surgiu um compilador da Linguagem de Descrição Formal Estelle.

No trabalho de [19] foi feita uma interface entre a saída do compilador Estelle de [14] e o gerador de Cadeia de Markov [18], compatibilizando a forma intermediária com o modelo orientado a objeto de [17] utilizado em [18].

Finalmente o trabalho resultante de [19] se destinou a integrar o Sistema de Auxílio como uma ferramenta de avaliação de desempenho.

Este histórico descrito exemplifica mais um de muitos casos de posterior integração de ferramentas e recursos que foram desenvolvidos em separado. Isto é muito comum e muitas vezes inevitável, como foi neste caso.

Com o passar do tempo a experiência mostrou que é muito interessante ter uma mesma base de dados, organizada num mesmo formato de estruturas, onde os recursos desenvolvidos para trabalhar tais estruturas pudessem ser facilmente reutilizados em outras situações de implementação com outros propósitos. Tal arquitetura de organização ainda possibilita a implementação de uma biblioteca de funções e tipos que podem ser utilizados sempre que necessário e até enriquecidos com novos tipos e funções.

Decidiu-se portanto implementar esta filosofia de organização aos futuros desenvolvimentos de novas ferramentas que irão se adicionar ao CAD de Estelle, surgindo a ideia de conceber uma arquitetura que servisse de plataforma de desenvolvimento de softwares.

A Plataforma de Software para Algoritmos de Busca em Protocolos Descritos em Estelle se propõe a auxiliar o desenvolvimento de ferramentas com uma filosofia de implementação que priorize uma maior facilidade de integração de tais ferramentas, sem a necessecidade de artifícios de adaptação e interfaceamento, o que nem sempre aconteceu. Por exemplo, em [19], existia a necessidade de parte do processamento ter que ser realizado no IBM grande porte e outra parte no IBM PC para a obtenção dos resultados.

Desta forma espera-se alcançar ferramentas mais acessíveis e difundidas entre os usuários, que possam ser naturalmente integradas ao Sistema de Auxílio ao Projeto de Protocolos, uma vez que forem desenvolvidas sobre os mesmos alicerces computacionais.

2.4.5. Planos de aplicação da Plataforma de Software

Muitas são as ferramentas que podem ser desenvolvidas, para auxiliar no estudo de sistemas de comunicação, que se baseiam em algoritmos de busca para

expansão de árvores. Entretanto existe um interesse em desenvolver também, ferramentas para integrar o Sistema de Auxílio, que possibilitem estudos na área de avaliação de desempenho, que também se utiliza de algoritmos de busca em suas análises.

Para se desenvolver uma ferramenta para avaliação de desempenho de sistemas de computação/comunicação é muito comum a utilização de modelos markovianos, onde modelos de filas markovianas representando uma rede de comunicação devem descrever :

* As filas do sistema,
* as rotas das mensagens,
* as taxas de chegada de mensagens,
* as conexões entre filas e
* as taxas de erro de transmissão dos canais de comunicação.

Uma boa representação matemática para este modelo seria uma matriz de transição de estados de uma Cadeia de Markov correspondente [26] e [27].

A utilização deste modelo matemático para os protocolos de comunicação permite que se obtenha medidas de desempenho a partir da Cadeia de Markov, de onde se obtém a Matriz de Transição de Estados do sistema.

2.4.6. Extensões Adicionadas na TDF

Caso porventura sejam futuramente adotados sistemas markovianos como modelo matemático dos sistemas de comunicação a serem estudados, isto implica na necessidade de inclusão de algum artifício para a descrição das taxas de disparo das transições, durante a descrição dos protocolos, que é feita com Estelle. Estas

informações não podem deixar de constar na especificação, seja de maneira direta ou indireta, para que depois possam ser extraídas, pois elas são uma característica inerente das Cadeias de Markov.

Os objetos de descrição do sistema de Estelle não fazem qualquer referência a essas taxas ou nem sequer ao menos às probabilidades, portanto foi acrescentada ao subconjunto de palavras reservadas de Estelle, que poderá ser utilizado pelas futuras ferramentas, a cláusula PROBABILITY, que deve ser declarada como uma variável em cada um dos módulos descritos e mencionada com o seu respectivo valor em cada uma das transições de todos os módulos da especificação. Estratégia semelhante foi implementada em [19].

Para possibilitar estudos de modelos markovianos, a Plataforma de Software possui capacidade de tornar visível ao algoritmo de busca nela implementado, o valor desta nova cláusula Estelle, no momento de disparo de cada uma das transições de estado, através de um recurso especial manipulado no compilador para Estelle.

Todos os detalhes da implementação deste e outros recursos oferecidos pela Plataforma de desenvolvimento de ferramentas, assim como o formato que deve ter sua respectiva declaração na especificação dos protocolos para posterior utilização pelo algoritmo, serão detalhados posteriormente.

2.4.7. Considerações Adotadas para as Ferramentas

Para exemplificar a utilização da Plataforma de Software, foram implementadas ferramentas de geração da árvore de alcançabilidade total de sistemas especificados em Estelle que permitem a exploração das características dos estados globais do protocolo testado de diferentes maneiras e sob vários aspectos.

Esta geração é feita de forma que toda ela é calculada, pois não faz parte do escopo deste projeto piloto o desenvolvimento de uma ferramenta muito sofisticada, com inclusão de heurísticas para minimizar o tempo de geração. Estas ferramentas desenvolvidas com o auxílio do ambiente computacional oferecido pela plataforma estão aptas a serem incrementadas posteriormente.

Além disso, o estudo de possíveis técnicas para podar ramos de pouco interesse da árvore de alcançabilidade deve ser feito quando conveniente, pois uma heurística útil em uma determinada ferramenta pode não ter validade em outra. Portanto o algoritmo de geração do grafo não utiliza heurísticas com esta finalidade.

Se levarmos em consideração que nosso sistema pode ser descrito como sendo um conjunto de componentes chamados de objetos do sistema, onde cada um destes objetos possui características próprias, os estados do sistema receberão o cunho de globais..

As características que descrevem um objeto qualquer no nosso modelo adotado são:

* capacidade de interagir entre si por troca de mensagens,

* cada objeto possui um conjunto de possíveis estados internos,

* os possíveis estados internos de um objeto podem se revesar com o transcorrer do "tempo",

* este "tempo" transcorre quando algum destes objetos gera algum evento ou quando recebe uma mensagem de um outro objeto.

Desta forma o estado global de um sistema já pode ser definido como um conjunto com os estados internos de cada um dos módulos, adicionados das respectivas mensagens que ainda não foram entregues.

No modelo utilizado, o tempo de reação de um objeto a uma mensagem recebida é considerado como nulo e a descrição formal de todos os objetos do sistema é feita com o auxílio da TDF Estelle acrescida da informação de probabilidade de disparo de cada uma de suas transições. Em cada ciclo de tempo, pode ser executada qualquer transição apta, de acordo com as normas de Estelle.

Para simplificar a modelagem, serão excluídos da análise sistemas com paralelismo assíncrono e para isso será considerado apenas um subconjunto das cláusulas de Estelle, assim como dos tipos de módulos. Este subconjunto, apesar de grande, não contará com os tipos de módulos SYSTEMPROCESS e PROCESS.

Desta maneira, indiretamente foi evitada a explosão de estados globais relacionados com o paralelismo assíncrono, diminuindo bastante com isso, a quantidade total de estados globais alcançáveis do sistema. Ainda não serão válidas as palavras reservadas ALL, EXIST e FORONE por não serem tratadas pelo compilador.

Não havendo o paralelismo assíncrono, vamos assumir que em cada ciclo de execução seja disparada somente uma transição, o que permite que o tempo possa ser classificado como progressivo e discreto.

Como as transições entre estados ocorrem em instantes definidos no tempo, podemos fazer uma analogia das árvores de alcançabilidade de estados globais geradas com Cadeias de Markov de tempo discreto, que podem modelar matematicamente os sistemas descritos em Estelle.

Como cada nó de nossa árvore de alcançabilidade representa um estado global de todo o sistema especificado no instante da transição, pode ser associada a uma Cadeia de Markov embutida.

Quando consideramos que as transições de estado podem ocorrer a qualquer instante, dizemos que a Cadeia de Markov é de tempo contínuo e o tempo entre as transições de estado possui uma distribuição exponencial.

O modelo de Estelle usando uma Cadeia de Markov de tempo discreto pode ser extendido a uma Cadeia de Markov de tempo contínuo [23], [24] e [25].

2.5. Conclusão

Neste capítulo foram apresentados os conceitos básicos necessários para a perfeita compreensão e desenvolvimento do trabalho proposto assim como a explicação de como estes conceitos foram utilizados e o porque desta utilização.

Foi feita uma explanação a respeito do surgimento das várias TDFs até chegarmos na descrição da técnica escolhida que deve fazer a formalização das especificações dos protocolos de comunicação em estudo.

Foram ainda justificadas as medidas tomadas para o desenvolvimento da Plataforma computacional para implementação de Algoritmos de Busca, assim como algumas considerações feitas para a confecção das duas ferramentas criadas.

Capítulo 3

3. Plataforma de Software para Desenvolvimento

3.1. Introdução

Com o desenvolvimento dos estudos das Técnicas de Descrição Formal, começaram a ser idealizadas ferramentas computacionais que, aproveitando esta nova possibilidade de padronização, surgiram com o objetivo de facilitar o trabalho dos engenheiros de sistemas, de projetar protocolos de comunicação.

Contribuindo com esta tendência, foi idealizado um Sistema de Auxílio ao Projeto de Protocolos descritos em Estelle, que é composto por várias ferramentas com este propósito. A maioria destas ferramentas se utiliza de uma entrada padrão denominada Forma Intermediária, e para agilizar o surgimento de novas ferramentas que oferecem recursos de estudo e implementação ao projetista de sistemas de comunicação, foi concebida uma Plataforma de Software para desenvolvimento de Algoritmos de Busca numa arquitetura organizada com este objetivo.

Neste capítulo são descritos o Sistema de Auxílio ao Projeto de Protocolos de Comunicação descritos em Estelle e a chamada Forma Intermediária (F.I.), nas seções 3.2 e 3.3, respectivamente, pois o conhecimento deles é fundamental para um perfeito entendimento da plataforma computacionacional a qual será apresentada em seguida na seção 3.4.

3.2. O Sistema de Auxílio

O Simulador de Protocolos [13] foi a primeira ferramenta do Sistema de Auxílio a ser desenvolvida, seguida do Compilador Estelle [14]. Neste simulador foram concebidos alguns procedimentos e estruturas de dados importantes que são obtidos a partir das informações disponibilizadas pela saída do Compilador Estelle, denominada de Forma Intermediária.

A primeira parte do Sistema de Auxílio [11] que se propõe a gerar a forma intermediária que servirá de entrada para as outras ferramentas possui uma arquitetura da seguinte forma:

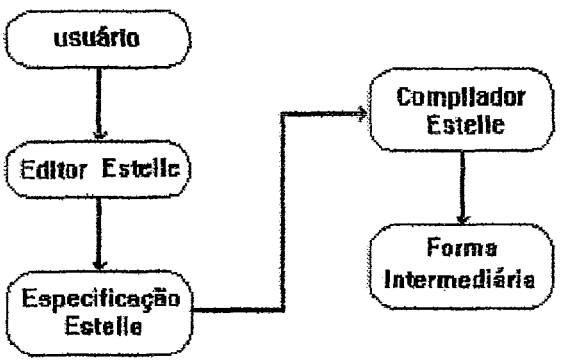

Figura 3.1 - Parte da arquitetura do Sistema de Auxílio que não depende da Forma Intermediária gerada pelo compilador.

Como muitas vezes as estruturas de dados desenvolvidas para cada uma das ferramentas do Sistema de Auxílio eram muito semelhantes e algumas vezes até iguais, percebeu-se que é muito conveniente que exista uma plataforma de desenvolvimento uniforme, possibilitando desta forma, que alguma parte da implementação de uma ferramenta sirva para ser aproveitada na confecção de

outras futuras ferramentas, facilitando a interface dos algoritmos implementados, com a F.I., ganhando tempo no desenvolvimento e diminuindo, se não evitando, o surgimento de trabalhos de integração entre ferramentas.

A outra parte do Sistema de Auxílio que engloba as ferramentas que dependem da Forma Intermediária como entrada, tem seu esquema descrito na fig. 3.2 mostrada a seguir.

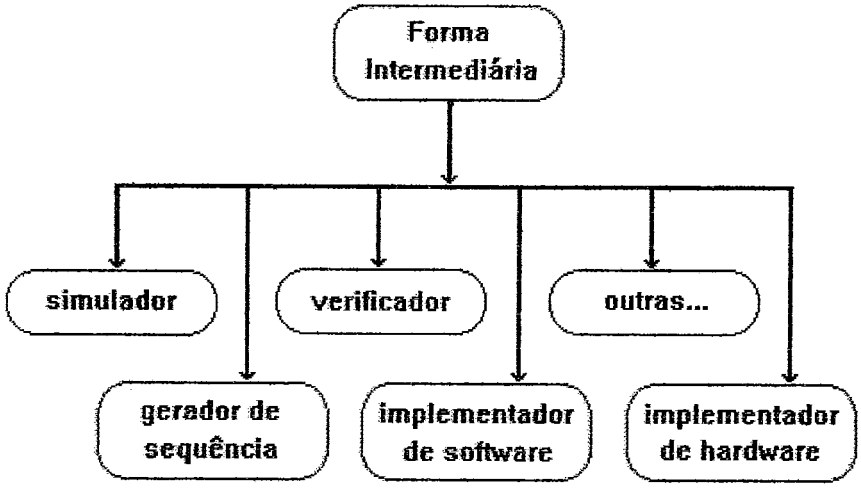

Figura 3.2 - Parte da arquitetura do Sistema de Auxílio que depende da Forma Intermediária gerada pelo compilador.

A) Editor [15]

Esta ferramenta combina recursos de edição gráfica com auxílio à sintaxe. Os recursos gráficos visam tornar a tarefa de especificação mais confortável, diminuindo o aparecimento de erros de sintaxe e reduzindo o tempo gasto na especificação de um protocolo de comunicação. Com este propósito, foi proposta uma forma gráfica para Estelle.

B) Compilador [14]

Este é responsável pela análise e deteção de erros na especificação do protocolo de comunicação. Após as devidas correções, esta ferramenta gera a Forma Intermediária da especificação Estelle, a qual será a entrada de vários outros elementos que compõem o sistema de auxílio ao projeto de protocolos.

C) Simulador [13]

Este programa executa a especificação Estelle das entidades dos protocolos, todas em uma mesma máquina, de forma seqüêncial, podendo portanto, testar os serviços e parte do desempenho dos protocolos de comunicação.

D) Implementador de Sotware [20]

Esta outra ferramenta é encarregada de gerar um código executável para uma dada máquina alvo, levando em consideração o seu respectivo sistema operacional.

E) Seqüênciador [12]

Baseado numa técnica para testar a conformidade do componente de controle de protocolos e serviços, o Gerador de Seqüências de Testes utiliza o conceito de uma única marca, denominada "seqüência única de entrada/saída", para cada estado da especificação do protocolo. As seqüências de testes geradas percorrem cada estado e suas transições.

F) Verificador [16]

O Verificador para Estelle tem a função de auxiliar o projetista de protocolo a corrigir a sua especificação a partir da definição de propriedades descritas com fórmulas de Lógica Temporal. A avaliação destas fórmulas pode ser feita com o emprego de técnicas de inteligência artificial para agilizar a expanção do grafo de alcançabilidade de estados.

G) Implementador de Hardware VLSI [21]

Este último componente do Sistema de Auxílio visa obter, de maneira automática, um esquema de máscaras de silício para posterior fabricação de circuitos integrados que sejam capazes de executar os serviços desejados do protocolo de comunicação especificado inicialmente.

Portanto o Sistema de Auxílio irá funcionar a partir do uso do Editor Estelle, que auxilia o usuário na geração da Especificação em Estelle do protocolo de comunicação que está sendo projetado. Esta especificação servirá de entrada para o Compilador Estelle que por sua vez irá gerar a Forma Intermediária (F.I.). Finalmente a FI será a entrada fundamental para todas as outras ferramentas que possibilitarão a conclusão dos trabalhos de simulação, verificação, seqüenciamento e outros. Caso todos estes obtenham resultados satisfatórios, será em seguida, utilizada uma das ferramentas de implementação, conforme for mais conveniente para a aplicação a qual se destina o protocolo projetado.

3.3. A Forma Intermediária

A chamada Forma Intermediária (F.I.) será apresentada nesta seção de maneira resumida, mas maiores detalhes são descritos em [14].

Como o próprio nome sugere, a F.I. é uma representação intermediária entre a especificação Estelle de um protocolo de comunicação e as respectivas estruturas de dados que comportam estas informações que ferramentas do Sistema de Auxílio, descritas na seção anterior, podem manipular através de procedimentos especiais .

A Forma Intermediária foi concebida para descrever as características de um sistema, apresentadas em duas partes a saber : a estática e a dinâmica. Estas duas partes são representadas por um esquema na fig. 3.3.

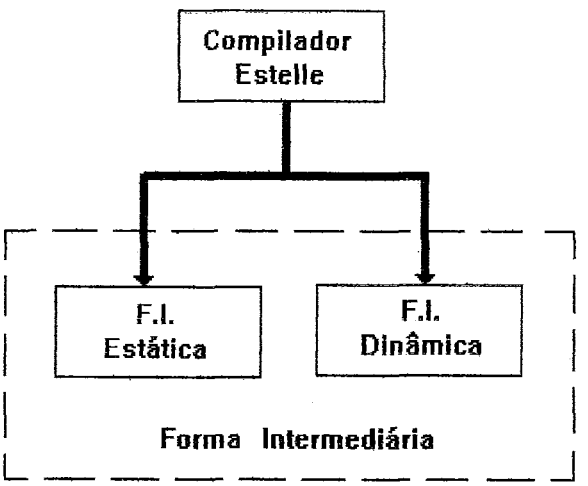

Figura 3.3 - Esquema da Forma Intermediária

A F.I. Estática

A parte da F.I. estática compreende todos os objetos de Estelle relacionados com sua topologia, vindos descritos em um arquivo ASCII de acordo com uma linguagem notacional bem definida.

A seguir, os conjuntos da F.I. estática que descrevem os objetos de descrição estáticos de uma especificação Estelle :

* O conjunto **Sb** contém a notação que identifica os corpos de módulo,

* O conjunto **BodySymbol** descreve a hierarquia dos corpos de módulos,

* Os conjuntos **Sinh_id, Ssyph_id, Ssyah_id** e **Sah_id** contêm os identificadores de cabeçalho de módulo,

* Os conjuntos **Sinit_id, Sinput_id, Sspont_id** e **Sot** contêm os identificadores de transições,

* O conjunto **Sms_id** contém os nomes dos estados dos módulos usados nas cláusulas FROM e TO,

* Os conjuntos **Seip_id** e **Siip_id** contêm o nome dos pontos de interação,

* Os conjuntos **Siqip_id** e **Scqiq_id** agrupam os identificadores dos pontos de interação de acordo com a política de utilização das filas FIFO,

* O conjunto **Sint_id** descreve os identificadores das interações definidas em canais dos corpos,

* O conjunto **Sm** mostra o nome das variáveis módulo para cada tipo de cabeçalho,

* Os conjuntos **Sdmdim** e **Sdipdim** contêm as dimensões dos vetores de variáveis módulo e pontos de interação,

* O conjunto **Transitions** mostra os valores mínimo e máximo da cláusula DELAY e o valor da cláusula PRIORITY de cada transição,

* Os conjuntos **Iptrans, Iprv_send** e **Iprv_receive** fornecem informações complementares relativas aos mecanismos de troca de mensagens,

* Os conjuntos **Sinput_state** e **Soutput_state** fazem a associação do estado de entrada de cada transição (cláusula FROM) ao respectivo estado de saída (cláusula TO) e

* O conjunto **Role** descreve as transições que podem ser enviadas e recebidas por cada IP.

A F.I. Dinâmica

É a outra parte da F.I. Esta é composta, fundamentalmente, por um arquivo com código em MODULA-2, que implementa a cláusula PROVIDED de Estelle, assim como a ação de suas transições (declarações BEGIN e END) e todo o resto de declarações em PASCAL da especificação.

Oferece ainda alguns procedimentos de ajuda às ferramentas do Sistema de Auxílio.

3.4. A Plataforma de Software para Estelle

Uma vez explicados a Forma intermediária e o Sistema de Auxílio, poderemos finalmente apresentar a Plataforma de Software para desenvolvimento de Algoritmos de Busca para o Sistema de Auxílio a Projetos de Protocolos de Comunicação que tem por finalidade fornecer condições ao projetista desenvolvedor facilidades para que seja possível o seguinte :

* Fazer o interfaceamento entre a saída do compilador Estelle e uma estrutura de dados organizada;

* Disponibilizar tal estrutura de dados de maneira encapsulada e documentada para utilização no desenvolvimento de outras ferramentas;

* Oferecer serviços de manipulação especiais para cada uma das estruturas de dados na forma de bibliotecas;

* Possibilidade de enriquecimento de tais bibliotecas com o acréscimo de novas estruturas e funções de manipulação, para posterior reutilização no desenvolvimento de outras ferramentas;

* Proporcionar uma padronização na implementação de futuras ferramentas de auxílio ao estudo e desenvolvimento de protocolos de comunicação.

* Evitar a necessidade do surgimento de trabalhos de adaptação, interfaceamento e integração entre ferramentas desenvolvidas ou mesmo das próprias ferramentas com o Sistema de Auxílio;

* Permitir o desenvolvimento de tais ferramentas e o incremento do Sistema de Auxílio para o ambiente do sistema operacional DOS;

* Dotar todas as ferramentas desenvolvidas sobre esta base computacional de características naturais que permitem uma imediata inclusão da mesma ao CAD de Estelle.

* Dar um suporte semelhante ao desenvolvimento de ferramentas para o Sistema de Auxílio, também para o ambiente do sistema operacional UNIX.

* Oferecer condições propícias de implementação, também aos desenvolvedores de ferramentas para estudos de análise de desempenho, através de estruturas de dados e serviços específicos com este propósito.

Utilizando-se o ambiente computacional proporcionado pela Plataforma de Software, foram implementadas duas ferramentas que podem auxiliar no estudo e desenvolvimento de protocolos de comunicação. Estas ferramentas, além de exemplificar a utilização dos recursos oferecidos por esta plataforma computacional, proporcionam os seguintes serviços:

* Prever todos os possíveis estados futuros para os quais um sistema descrito em Estelle pode evoluir;

* Determinar os caminhos percorridos pelo algoritmo para alcançar os possíveis estados globais futuros do sistema descrito;

* Fazer avaliações probabilísticas da evolução do sistema em teste para seus estados globais futuros;

* Descobrir falhas de concepção de projeto do protocolo de comunicação e descrever quando o sistema evolui para um estado global terminal ou DEADLOCK;

* Fazer a visualização detalhada de todas as características de cada estado global do sistema quanto a todos os seus aspectos, tais como: o estado local de cada entidade do sistema, suas respectivas filas, hereditariedade, etc;

* Evitar que um protocolo de comunicação tenha que ser implementado para que, só então, durante a fase dos testes de validação, sejam detectados problemas de funcionamento ou insatisfação de performance.

Portanto, o objetivo principal da Plataforma de Software, e mais ainda, do Sistema de Auxílio, é facilitar a confecção dos projetos de protocolos, através do desenvolvimento de ferramentas que determinam, etapas padronizadas de procedimento para implementação, evitando assim, esforços e tempo desnecessários, e otimizando custos e recursos humanos.

Estas duas ferramentas são de aplicação genérica, apesar de terem sido desenvolvidas com um propósito específico, podendo ser utilizadas em uma infinidade de áreas do conhecimento nas quais haja algum tipo de mudança de comportamento a partir da troca de informações ou mesmo somente de simples atitudes.

A maior dificuldade de se fazer uma especificação costuma ser conseguir ter o conhecimento de todas as atitudes de cada uma das entidades que compõem o sistema a ser estudado, em função dos possíveis eventos que podem vir a acontecer com as entidades. Tendo este conhecimento, a modelagem do problema fica muito coerente com a realidade e torna possível que sejam feitas análises e previsões bem precisas de qualquer sistema, seja ele computacional, econômico, psicológico ou qualquer outro.

3.4.1. A Implementação

Como foi explicado anteriormente, uma das estratégias de procedimento a ser adotada na concepção da Plataforma de Software planejada desde a fase de escolha da metodologia de desenvolvimento, foi de utilizar a mesma linguagem de programação adotada na confecção do Compilador para Estelle para facilitar a compatibilidade e evitar links entre linguagens diferentes.

Desta maneira, fica uma orientação no sentido de que a linguagem de programação que desenvolvedores de ferramentas para o Sistema de Auxílio devem preferencialmente utilizar é MODULA-2 também.

O desenvolvimento desta plataforma foi baseado nos recursos oferecidos pelo Compilador Estelle e em duas outras ferramentas, o Simulador e o Verificador, já disponíveis no CAD de Estelle nas versões para DOS.

Módulos com as estruturas de dados de interesse, que basicamente estão aptas a receber as informações da F.I. do Compilador, associadas às respectivas funcões e procedimentos de manipulação destas estruturas, foram adicionados à outras estruturas e procedimentos que permitem o desenvolvimento de ferramentas baseadas em algoritmos de busca, de interesse inclusive para análise de desempenho, foram organizados em bibliotecas e devidamente documentados para posterior utilização.

Para o ambiente dos microcomputadores IBM PCs e compatíveis foi feita uma estrutura de implementação organizada segundo uma filosofia de overlays, que está apta a ser facilmente adaptada resultando em novas ferramentas para o Sistema de Auxílio. Uma nova ferramenta a ser implementada deve ter seu respectivo algoritmo gerador de resultados num overlay particular para que se adapte naturalmente a todo o resto da plataforma de desenvolvimento, conforme foi feito na criação de duas ferramentas que são apresentadas ainda neste capítulo.

Como atualmente existe um grande interesse em tornar operacional um outro Sistema de Auxílio, também para o ambiente das SPARC Stations da SUN, foi feito todo um trabalho de conversão e adaptação de todas estruturas de dados assim como seus respectivos procedimentos de manipulação, para este outro ambiente.

Para realizar tal tarefa, foi modificada toda a filosofia de implementação da arquitetura da Plataforma de Software, que na versão para desenvolvedores em UNIX, não se utiliza da técnica de overlays.

Além disso, foi necessário implementar bibliotecas de conversão de funções para adaptar todo o código aos recursos disponíveis no ambiente UNIX, pois havia incompatibilidades de software.

3.4.1.1. Alguns aspectos da linguagem MODULA-2

A linguagem de implementação MODULA-2 foi a eleita para a implementação da plataforma de desenvolvimento e possui algumas características de sua estrutura que devem ser apresentadas para um melhor entendimento do próprio desenvolvimento e, posteriormente, da própria utilização da plataforma na confecção de ferramentas baseadas em algoritmos de busca.

Esta linguagem subdivide as informações dos programas desenvolvidos em alguns arquivos de diferentes extensões.

* arquivos.MOD,
* arquivos.DEF,
* arquivos.OBJ,
* arquivos.REF,
* arquivos.SYM,
* arquivos.OVL e
* arquivos.EXE.

Nos arquivos .MOD residem as instruções, propriamente ditas, do corpo principal do programa em desenvolvimento. São os módulos do sistema.

Como esta linguagem de programação possui a capacidade de ser desenvolvida de maneira modular para facilitar e melhor organizar a implementação, existem os arquivos .DEF, que são responsáveis pelo intercâmbio entre os módulos. Cada módulo possui um respectivo arquivo de definição que descreve todos os recursos (variáveis globais e procedures) que ele importa dos outros módulos e quais recursos ele torna diponível aos outros módulos do sistema, para que estes também possam fazer suas importações.

A compilação de um arquivo .MOD gera dois outros arquivos, o .OBJ e o .REF, chamados respectivamente de arquivo objeto e de referência. Os arquivos de referência têm a finalidade de servir de entrada ao depurador da linguagem, sendo que, uma vez corrigidos todos os erros da compilação de um módulo, seu respectivo arquivo .REF se torna desnecessário.

O resultado da compilação de um arquivo .DEF é um respectivo arquivo .SYM que, juntamente com o respectivo .OBJ, será utilizado no processo de linkedição para gerar finalmente o arquivo de extensão .EXE, que é o programa executável.

No caso de impossibilidade de, a máquina em que se está realisando o desenvolvimento, oferecer maior capacidade de memória RAM, que seja suficiente para carregar todas as informações necessárias para rodar o programa, existe a possibilidade de subdividir o processamento do programa desenvolvido em "pedaços" para serem executados um de cada vez. Estes "pedaços" de código a serem executados de maneira intercalada são os arquivos de extensão .OVL, chamados de overlays.

Este recurso de subdividir o processamento do programa foi utilizado no desenvolvimento da ferramenta para a versão DOS, o que já não aconteceu durante

o desenvolvimento da versão para UNIX, pois o hardware das estações de trabalho torna isto desnecessário. No software para PCs foram gerados seis overlays distintos que foram denominados:

* ABORTO,	* PASSOAPA,
* INFORMAC,	* PROPRIED e
* INICIADO,	* RESULTAD.

Após um overlay ser executado, ele é descarregado da memória para que outro overlay, que se faz necessário para a continuação do processamento do sistema, seja carregado. Este procedimento é repetido quantas vezes forem necessárias durante a execução de um teste.

Já na versão UNIX da ferramenta, esta estratégia não se faz necessária devido à maior capacidade de memória, velocidade de processamento e recursos multitarefa oferecidos pelo sistema operacional UNIX. Portanto cada overlay da versão DOS irá se transformar num módulo com o seu respectivo arquivo de definição na versão UNIX.

3.4.1.2. A Implementação da Cláusula de Probabilidade

Como a TDF Estelle não possui um suporte natural para descrever as características dos sistemas em geral, quanto à probabilidade de ocorrência de seus eventos, se fez necessária a inclusão de artifícios para possibilitar tais descrições nas especificações dos protocolos de comunicação feitas em Estelle.

Foi idealizada a criação de uma variável com um nome pré-escolhido, para representar o valor da probabilidade dentro das especificações.

Sabendo-se o nome desta variável a priori, seria mais fácil fazer uma varredura na especificação, em busca do valor da probabilidade de cada transição, caso contrário, o nome da variável que contivesse esta informação de interesse, deveria ser passado para o sistema, durante o tempo de sua execução.

Se o nome de tal variável não fosse fixo, faria com que, além de não dar continuidade ao aspecto da descrição do protocolo ser padrão, tornaria a inclusão desta capacidade de descrição à TDF Estelle, mais complexa.

Desta forma surgiu uma nova cláusula Estelle que passou a fazer parte do conjunto de palavras reservadas da TDF em uso.

O usuário das ferramentas desenvolvidas neste trabalho, que utilizam tal recurso oferecido pela plataforma, é orientado portanto, a se referir ao valor da probabilidade de disparo de cada transição na especificação, utilizando várias variáveis que devem ser declaradas, uma em cada corpo de tipo de módulo da especificação, sendo que todas elas possuem, obrigatoriamente, o mesmo nome, em qualquer especificação, para que esta representação se torne padrão.

Para facilitar o seu entendimento por parte de quem lê a especificação formal, foi determinado que tal variável deve ser denominada de PROBABILITY, deixando o seu significado no sistema, intuitivo dentro da especificação.

Ao gerar a F.I., o compilador disponibiliza um procedimento que é exportado pela sua parte dinâmica, que permite que os valores de qualquer variável, expressão ou parâmetro declarados na especificação, sejam observados e alterados.

Este procedimento se chama *Interface Variáveis* e é utilizado pelo Gerador para associar um valor de probabilidade a cada uma das transições da especificação, através do uso de diretivas para o compilador, que tornam possível a visualização da palavra reservada PROBABILITY, que foi incluída no conjunto de cláusulas Estelle, especialmente com esta finalidade.

A seguir é mostrada a maneira de se utilizar destas diretivas.

{$Verificar VAR nome_var}

Esta diretiva acima permite tornar o valor da variável *nome_var*, visível aos algoritmos de busca embutidos na arquitetura da plataforma.

Com esta outra, pode-se alterar, em tempo de execução, o valor da variável.

{$Afetar VAR := PROBABILITY}

As variáveis que podem ser usadas nessa interface podem ser do tipo BOOLEAN, ARRAY OF CHAR, CHAR, REAL e INTEGER.

As ferramentas de geração de árvores de alcançabilidade que exemplificam a utilização da plataforma, na sua fase de inicialização, fazem o registro de todas as variáveis, expressões e parâmetros declarados pelas diretivas do tipo $Verificar a fim de que possam ser usados na computação de estados globais.

Tais variáveis, expressões e parâmetros passam a ser identificados pelas estruturas de dados especialmente desenvolvidas, através do número das declarações de suas diretivas, que é determinado pela ordem em que aparecem no texto. Portanto a primeira diretiva do tipo $Verificar, a aparecer na especificação, recebe o número 1, a segunda o número 2, e assim por diante. O mesmo acontece com a diretiva $Afetar, só que esta definirá uma outra numeração.

A seguir uma exemplificação da maneira que o usuário deve proceder para dotar suas especificações formais da informação de probabilidade das suas respectivas transições de disparo.

```
BODY nome_corpo FOR nome_tipo_módulo

VAR
        PROBABILITY : REAL;
        {$Verificar VAR PROBABILITY}

INITIALIZE BEGIN
        PROBABILITY := 1;
END;

TRANS
        .
        .
        .
```

Um procedimento recomendável, é sempre inicializar todas as variáveis de qualquer sistema, com valores coerentes, para evitar erros provocados por valores "sujos" que podem estar presentes na área de memória alocada para cada variável, durante a inicialização.

No caso da probabilidade, este procedimento deve ser feito dentro da cláusula INITIALIZE, sem esquecer que seu valor pode ser um real *n* qualquer compreendido entre 0 (zero) e 1 (um) inclusives.

Feito isso, o valor da variável deve ser setado dentro de cada uma das transições, pois se não, o gerador de árvores de alcançabilidade probabilísticas vai considerar seu valor como sendo o da transição anterior.

A atribuição de um valor de probabilidade de disparo às transições é feita da seguinte forma.

```
TRANS
        NAME transição1;
        BEGIN
                PROBABILITY := n1;
        END;

TRANS
        NAME transição2;
        BEGIN
                PROBABILITY := n2;
        END;
```

Para auxiliar na atribuição de valores de probabilidade para cada transição, foi implementado um serviço especial de verificação do valor da soma de tais probabilidades relacionadas com transições que saem de um mesmo estado global. Caso esta soma não seja igual a 1 (um), o usuário é informado de em qual estado global do sistema isto não ocorreu, para que possa ser facilmente corrigido.

3.4.2. A Arquitetura da Plataforma de Software

O esquema funcional da plataforma de desenvolvimento foi organizado em grupos de módulos que se relacionam segundo o esquema descrito pela fig. 3.4.

Apesar de não fazerem, efetivamente, parte da estrutura implementada para a plataforma, foram acrescentados os blocos: *Editor Estelle*, *Compilador Estelle*, *F.I. Estática* e *F.I. Dinâmica*, para tornar mais claro o funcionamento da plataforma, assim como do próprio Sistema de Auxílio, como um todo, e mostrar de que maneira os algoritmos de busca se encaixa no Sistema de Auxílio para

resultar no surgimento de uma nova ferramenta dentro do CAD de Estelle. Na fig.3.4, estes blocos estão diferenciados com um duplo traço à esquerda.

Após a geração da Forma Intermediária, suas partes Estática e Dinâmica são consultadas para a formação de algumas das estruturas de dados que serão utilizadas pelas ferramentas. Os módulos de implementação do Simulador aproveitados no Verificador e herdados pela Plataforma de Software, estão relacionados com as estruturas de dados referentes aos objetos de Estelle (*Dados Estáticos* e *Dados Dinâmicos*) e o módulo que executa suas primitivas (*Executor das Primitivas*).

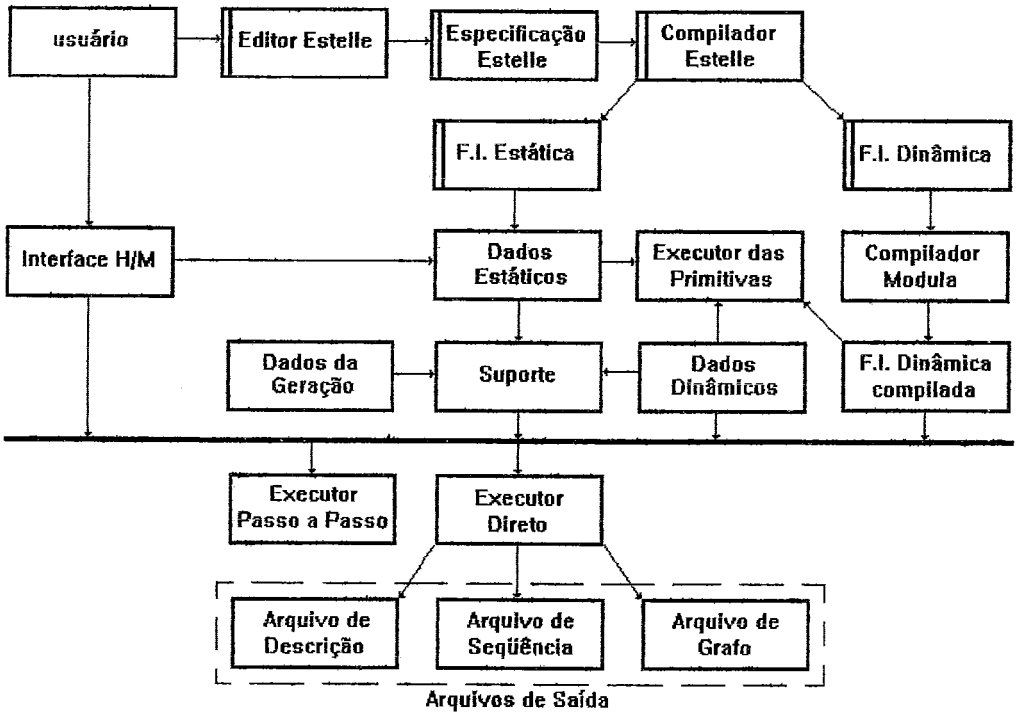

Figura 3.4 - Arquitetura geral de funcionamento da Plataforma de Software.

Os dados estáticos referem-se à topologia da especificação Estelle, sendo carregados pelas informações do arquivo da F.I. estática e não mudam durante o processo de geração dos estados globais do protocolo.

Os dados dinâmicos entretanto, mudam durante este processo, tendo seu valor inicial estabelecido na parte de inicialização da especificação e, a partir daí, variam constantemente com o disparo de cada transição.

Entre o disparo de uma transição e outra, se estabelece uma situação instantânea chamada de contexto da especificação, que constitui justamente o estado global atual em análise no Gerador de Estados.

Além das estruturas de dados, existe ainda um módulo criado para dados referentes ao funcionamento do próprio Gerador, que são responsáveis pela execução da geração dos estados globais do protocolo testado.

O sistema foi concebido de modo a oferecer os serviços das duas ferramentas de geração:

* Passo a passo - Consiste de um acompanhamento detalhado de cada estado global do protocolo, por parte do usuário, que pode ir disparando uma transição por vez, a seu critério, e desta forma, ir percorrendo a Árvore de Alcançabilidade do protocolo.

* Direto - Gera toda a Árvore de Alcançabilidade do protocolo de uma vez só e aproveita para fazer a análise de algumas das propriedades validadas normalmente em Redes de Petri que serão explicadas na seção 3.4.2.1. Os resultados gerados por este tipo de serviço são mostrados através de três arquivos de saída.

Este serviço é realizado pelos Executores da Geração (*Executor Passo a Passo* e *Executor Direto*). Um destes executores grava seus resultados obtidos em

arquivos de saída que têm seus nomes definidos pelo usuário no início da execução da ferramenta de geração.

Conforme pode ser verificado na fig. 3.4, são três os arquivos gerados pela ferramenta e estes são listados a seguir.

* Descrição dos Resultados - Em forma de texto e com extensão .DES, contém informações tais como: a razão do final do teste, a conclusão da análise do protocolo, o número de estados globais gerados, o tempo gasto e a descrição de algumas das propriedades do protocolo.

* Seqüência de Disparos - Em forma de texto e com extensão .SEQ, é o responsável por mostrar a evolução da análise através de uma Árvore de Alcançabilidade do sistema, registrando de qual módulo foi disparada qual transição e em que ordem foram disparadas, seguidas de suas respectivas probabilidades de disparo.

* Grafo de Execução - Arquivo do tipo binário com extensão .GRF, armazena a fila de estados globais, gerada durante um teste de geração, contendo todos os dados dinâmicos de Estelle.

O apêndice B mostra alguns exemplos de protocolos testados, assim como suas respectivas especificações em Estelle e seus arquivos de saída.

Apenas os arquivos de descrição e os de seqüência são listados no apêndice, pois, conforme foi explicado, o arquivo de extensão .GRF é binário e somente a ferramenta, através da plataforma, pode entender suas informações, que descrevem os estados globais obtidos durante a geração, e são utilizadas para fazer a exposição destes estados ao usuário através da interface da plataforma com o exterior. Portanto ficou sem propósito a sua presença nas listagens do apêndice.

O arquivo que mostra as seqüências de disparo da geração da Árvore de Alcançabilidade apresenta suas informações com o seguinte formato geral:

(Origem) -Modulo **n** -Transição **nome [d1,d2]** -Prob **valor** --> **(Destino)**

Onde:

* Origem - O estado global inicial,
* n - A identificação do módulo Estelle ativo,
* nome - O nome de qual transição do módulo *n* foi disparada,
* d1 e d2 - Os valores máximo e mínimo da cláusula DELAY,
* valor - A probabilidade de disparo da transição disparada e
* Destino - O estado global final.

A seguir serão mostradas algumas linhas do arquivo .SEQ, obtido pela ferramenta que implementa a geração direta das árvores de alcançabilidade, após o teste com o protocolo da Exclusão Mútua.

(2) -Modulo 1 -Transição RECURSO1 -Prob 0.50000 --> (3)

Esta descreve que o sistema estava no Estado Global 2 e evoluiu para o Estado Global 3, após o Módulo 1 da especificação se manifestar e disparar a transição de nome RECURSO1, com uma probabilidade de 0,5 que pode ser expressa com uma precisão de até 5 casas decimais.

A geração da Árvore de Alcançabilidade é feita por uma busca em profundidade na árvore de alcançabilidade do sistema sem levar em consideração o tempo, para obter todos os estados alcançáveis.

Quando, a partir de um estado global, é disparada uma transição que leva o sistema para um estado já percorrido, o algoritmo do *Executor Direto* realiza um retorno ao estado anterior para tentar disparar uma transição que leve o sistema a um estado global ainda não visitado.

Este procedimento se denomina BACKTRACK e é mostrado no exemplo que segue.

(14) -Modulo 1 -Transição RECURSO2 -Prob 0,50000 --> (10)

** BACKTRACK TO (14) **

(14) -Modulo 2 -Transição PROCE2 -Prob 0,50000 --> (15)

Quando após um BACKTRACK, o estado global para que o sistema retornou, não possui mais nenhuma transição de qualquer módulo que ainda não foi disparada, o algoritmo retorna o sistema ao estado global anterior ao atual, e indica este procedimento da seguinte forma:

** BACKTRACK TO (27) **

(27) --VOLTA PARA--> (26)

(26) -Modulo 3 -Transição PROCE4 -Prob 0,50000 --> (28)

3.4.2.1. Descrição dos Módulos

Devido a grande quantidade de módulos que compõem a plataforma, foi concebida uma arquitetura em que estes módulos são subdivididos em grupos que contém módulos com objetivos semelhantes, para tornar mais fácil a compreensão do funcionamento interno da plataforma de desenvolvimento e sua posterior utilização (vide fig. 3.4).

Nesta seção, cada um destes grupos terá sua função na arquitetura, devidamente explicada, assim como o objetivo de seus módulos integrantes.

A implementação dos dados referentes à Estelle, assim como todos os outros, foi feita segundo uma filosofia que consistiu no encapsulamento destes dados. Todos as novas estruturas de dados criadas para o desenvolvimento de outas ferramentas devem seguir o mesmo padrão.

De acordo com as características estruturais de MODULA-2, o encapsulamento de cada tipo ou grupo de dados se deu por meio da sua implementação em cada módulo, mostrando na sua parte de definição (arquivos com extensão .DEF) as operações possíveis e disponíveis de serem realizadas sobre esses dados.

A maior parte dos objetos referentes aos dados manipulados, possui um aspecto variável a respeito de seu comprimento, isto é, a quantidade de informação a ser tratada pode mudar de especificação para especificação ou mesmo entre duas computações distintas dentro de uma mesma especificação.

Por exemplo, o número de corpos definidos muda de uma especificação para outra e a quantidade total de instâncias de módulo pode variar entre dois estados globais. Sendo assim, esses dados foram implementados com variáveis dinâmicas, formando filas encadeadas.

Apesar de a plataforma já disponibilizar muitos módulos que contém a implementação de filas encadeadas com dados de Estelle e outros com dados específicos das ferramentas de geração de estados globais, existem situações em que o algoritmo de busca em desenvolvimento requer novas estruturas de dados e procedimentos para executar, nessas filas, manipulações específicas ao seu interesse, que devem ser acrescentados à Plataforma para uma possível posterior utilização.

A seguir, estes e os demais módulos da plataforma terão suas respectivas finalidades na arquitetura descritas, separados em grupos que foram apresentados na fig. 3.4.

O grupo dos "Dados Estáticos"

Constituído por módulos que armazenam, em filas encadeadas, as informações lidas do arquivo da F.I. estática, cada módulo do grupo dos dados estáticos registra as informações de cada um dos conjuntos de notações da F.I. estática.

A relação destes módulos com os conjuntos da F.I. descritos durante a seção 3.3 é feita na tabela 3.1.

* BodySymbol - Contém o nome completo do corpo.

* SH - Implementa os identificadores de cabeçalhos (HId) e os relaciona com os corpos. Se refere à cláusula MODULE.

* IPTrans - Relaciona os identificadores dos pontos de interação (IPs) com identificadores de transição (TId).

* Role - Mostra o papel das primitivas nos canais, dizendo se elas são de entrada ou de saída.

* SB - Faz uma referência numérica (BId) aos corpos da especificação. Registra a informação da cláusula BODY FOR.

* STrans - Associa os identificadores de transição aos nomes dados às mesmas definidos na espcificação, relacionando as cláusulas TRANS e NAME.

* SInputState - Se refere ao estado inicial local de cada transição descrito pela cláusula FROM.

* SOutputState - Se refere ao estado final local de cada transição descrito pela cláusula TO.

* SState - Implementa os identificadores numéricos (SId) dos estados locais, relacionando estes com os seus respectivos nomes definidos na especificação pela cláusula STATE.

* SIntPt - relaciona os identificadores de IPs com seus respectivos cabeçalhos.

* SDipdim - Contém a dimensão do vetor de IPs de mesmo tipo para um mesmo cabeçalho.

* SInteraction - Relaciona o nome das primitivas com o canal de comunicação entre IPs

* SM - Registra as instâncias que podem vir a ser criadas durante a execução do sistema especificado, através das informações da cláusula MODVAR.

* SDmdim - Contém a dimensão do vetor que registra as instanciações de um mesmo módulo, definidas em MODVAR.

* Transitions - Registra as informações descritas na especificação através das cláusulas DELAY e PRIORITY.

A declaração de todos os tipos de dados de Estelle são registradas no módulo SType.

O overlay INICIADOR, citado pela primeira vez na seção 3.4.1.1, carrega as filas encadeadas implementadas nos módulos dos dados estáticos durante a fase de inicialização de qualquer ferramenta, sendo que, após esta carga, seus conteúdos não mais se alteram durante toda a análise de uma especificação.

MÓDULOS DO GERADOR	CONJUNTOS DA F.I. ESTÁTICA
BodySymbol SH	BodySymbol Sah_id
IPTrans	Iptrans
Role	Role
SB	Sb
STrans	Sinit_id Sinput_id Sspont_id Sot
SInputState	Sinput_state
SOutputState	Soutput_state
SState	Sms_id
SIntPt	Seip_id Siip_id Siqip_id
SDipdim	Sdipdim
SInteraction	Sint_id
SM	Sm
SDmdim	Sdmdim
Transitions	Transitions

Tabela 3.1 - Módulos da plataforma que implementam os conjuntos de dados de Estelle.

O grupo dos "Dados Dinâmicos"

Este grupo de módulos implementa os dados dinâmicos de Estelle nas ferramentas. A seguir será explicada a finalidade de cada um dos módulos que compõem o grupo dos Dados Dinâmicos.

* ModRef - Contém a implementação do conjunto de referências das instâncias de módulo (MRef) associadas às referências das suas respectivas instâncias de módulo pai, como também ao índice e ao identificador de suas variáveis módulo.

* SChild - Contém o conjunto que relaciona cada referência de instância de módulo às suas instâncias filhas que foram criadas.

* SBody - Implementa o conjunto de dados que relaciona o identificador estático de corpo a cada referência de instância de módulo existente.

* SExportVariable - Contém a estrutura de dados das variáveis exportadas (são variáveis declaradas no cabeçalho do módulo filho que podem ser compartilhadas com os pais), associadas a cada referência de instância de módulo.

* SLocalVariable - Contém a implementação das estruturas de dados referentes às variáveis de corpo associadas às referências de instâncias de módulo.

* SParameterVariable - Contém a implementação das estruturas de dados referentes aos parâmetros de módulos associados às referências de suas instâncias.

* Interactions - Implementa as estruturas de dados referentes aos parâmetros das primitivas de comunicação (interações) recebidas pelos módulos associados às referências de suas instâncias.

* SIPointRef - Responsável pelo armazenamento das referências dos pontos de interações dos módulos instanciados. Tais referências são relacionadas à

referência de suas instâncias de módulo, ao índice do vetor de instâncias de IPs e aos identificadores estáticos desses pontos na especificação.

* SAttached - Contém a implementação do conjunto dos pontos de interação ligados por meio de um ATTACH, registrando as ligações entre módulos pais e filhos.

* SAttach - Assim como o anterior, este módulo possui o conjunto dos pontos de interação ligados por um ATTACH, entretanto, são armazenados aqueles pontos no final das ligações, ou seja, os pontos extremos de vários ATTACHs realizados sucessivamente.

* SConnect - Este também implementa o armazenamento das referências de pontos de interações ligados, sendo que, desta vez, por meio de um CONNECT, registrando as ligações entre módulos com mesma hierarquia.

* SQueueRef - Este contém o conjunto das referências das filas FIFO do tipo INDIVIDUAL associadas às respectivas referências dos seus pontos de interação.

* SQRef - Contém a implementação das filas FIFO que são identificadas pelas suas referências criadas pelo módulo anterior, contendo as filas de primitivas de comunicação (mensagens).

* SToken - Registra o conjunto dos estados com ficha de cada instância, através dos identificadores estáticos dos estados associados às referências das instâncias de módulo, indicando os estados locais ativos de cada instância.

* SSpontTrans - Armazena as transições espontâneas sensibilizadas(que não possuem WHEN) de cada instância , relacionadas com os valores de suas temporizações, pois estes variam a cada disparo de transição.

* TransDEdisparo - Possui a implementação de dois conjuntos de informações com características específicas das ferramentas que exemplificam a utilização da plataforma: o conjunto das instâncias de transições disparadas de um

determinado estado global, e o conjunto das instâncias de transições disparáveis de um determinado estado global.

O grupo dos "Dados da Geração"

As estruturas de dados, específicas das ferramentas implementadas, que controlam o tipo de análise a ser realizada, estão implemenadas em cinco módulos :

* EstadosOperandos - Responsável pela definição das estruturas de dados que constituem os estados globais operandos. Estes estados, diferentemente dos dados de Estelle, foram estruturados em ARRAYs, porém seus elementos são RECORDs com alguns campos com ponteiros para filas encadeadas. São estados globais que aparecem como operandos nas fórmulas de Lógica Temporal, utilizados apenas pelo Verificador.

* Expressoes - Implementa a fila cujos elementos são os identificadores das diretivas declaradas na especificação para observar os valores de variáveis, expressões ou parâmetros, definindo o número das expressões ou variáveis que ficaram visíveis ao algoritmo.

* ResetaExpressoes - As expressões, variáveis e parâmetros contidos na fila de dados implementada pelo módulo *Expressoes* são aqueles válidos para a geração de estados globais, e o módulo *ResetaExpressoes* contém os procedimentos que selecionam quais variáveis e expressões utilizadas pelas diretivas do compilador, devem ser consideradas na formação dos estados globais.

* EstadosGlobaisGerados - Implementa todos os procedimentos relacionados com a geração e manipulação da fila de estados globais que constitui a saída em arquivo do grafo de execução do protocolo especificado. Portanto os procedimentos deste módulo realizam operações sobre o arquivo .GRF.

* DadosDAverificacao - Implementa records com ponteiros para todos os dados dinâmicos de Estelle que formam um estado global, e o restante dos dados das ferramentas de geração que, embora muitos, são estruturas com pouca complexidade. As variáveis que implementam este grupo de dados são exportadas diretamente para consulta e/ou alterção em outros módulos, não tendo encapsulamento como nas outras estruturas de dados, pois são simples, embora necessárias em quase todos os módulos.

O grupo do "Executor das Primitivas" de Estelle

Este bloco é composto por um módulo, o *SPrimitives*, que é responsável pela implementação dos procedimentos executores das primitivas de Estelle (INIT, WHEN, OUTPUT, CONNECT, ATTACH, DISCONNECT, ...).

Estes procedimentos foram desenvolvidos com base na definição das primitivas, fazendo todas as manipulações necessárias nos módulos das estruturas de dados dinâmicos de Estelle.

Além de inicializar o módulo da especificação através do procedimento *InitEspec* que executa o BEGIN END do INITIALIZE, dispara as transições através do procedimento *Fogo* que executa todos os BEGIN END dos TRANS.

Os grupos dos "Executores" da Geração

Este grupo de módulos implementa os dois algoritmos de busca que resultam nas Árvores de Alcançabilidade dos protocolos testados. Os módulos que compõem as estruturas executoras dos algoritmos são os seguintes :

* PassoApasso - Este módulo cria e apresenta ao usuário o Menu Passo a Passo, cujas opções permitem: A exposição das transições disparáveis para que o

usuário possa escolher uma a uma, a seu critério, para execução; A exposição das variáveis identificadas pelas diretivas do tipo $Afetar, declaradas dentro da especificação, para que o usuário possa modificá-las; E a exposição do contexto atual da especificação.

Desta maneira o usuário pode percorrer a Árvore de Alcançabilidade da maneira que quiser, possibilitando que sejam feitos estudos mais específicos do comportamento do protocolo.

* Propriedades - Este módulo gera uma Árvore de Alcançabilidade do protocolo especificado e ainda faz a verificação de algumas das propriedades gerais do sistema, tais como: Ausência de Bloqueio, Reinicialização e Limitação.

A geração da Árvore de Alcançabilidade do protocolo em análise é feita por meio de uma expansão total do grafo, a partir do estado global raíz, formado pela inicialização do módulo da especificação Estelle. Todos os outros estados globais possíveis de serem alcançados, vão sendo obtidos através do disparo sucessivo de todas as transições habilitadas para execução em todos esses estados.

A expansão da árvore de alcançabilidade é feita por meio de uma busca em profundidade.

Durante a expansão da árvore de alcançabilidade, se for obtido um estado global com o conjunto de transições disparáveis vazio, a verificação das propriedades termina neste ponto, indicando Bloqueio.

Por outro lado, no caso da árvore de alcançabilidade ser esgotada, chega-se à conclusão que a especificação é Limitada e Sem Bloqueio, sendo possível a partir de então, fazer a análise da reinicialização do protocolo.

A conclusão de que a especificação é Não Limitada, é atingida quando a árvore de alcançabilidade não se esgota devido ao aborto do usuário ou ao esgotamento de recursos.

A análise da reinicialização do protocolo é feita diretamente sobre a estrutura de dados que constitui os estados globais da árvore de alcançabilidade, usando, principalmente, um campo desta estrutura que indica se, pelo menos um sucessor direto do estado global, é o raiz. Esta análise consiste basicamente em observar se, para cada estado global gerado, existe pelo menos uma seqüência de sucessores que conduza ao estado global raiz. Se isto for constatado para todos os estados globais gerados, o protocolo é classificado como Reinicializável.

O grupo dos módulos de "Suporte"

O grupo de suporte possui as mais variadas funções que são utilizadas por vários outros grupos da arquitetura. Este grupo é composto pelos seguintes módulos:

* GravaResultados - Responsável pela geração do arquivo de descrição dos resultados.

* CarregaResultado - Utilizado na interface com o usuário durante a definição do estado inicial ou a apresentação de algum estado global presente no arquivo do grafo de execução, pois sua função é de carregar para a memória um determinado estado global desse arquivo, a fim de constituir o estado inicial ou simplesmente ser mostrado ao usuário na tela.

* Stype - Possui a definição de todos os tipos de objetos de Estelle, tais como: o identificador de corpo, a referência de instância de módulo, entre outros.

* DadosAux - Contém a definição de diversas estruturas auxiliares que são utilizadas por vários outros módulos com tarefas diferentes.

* TabRef - Implementa uma fila encadeada utilizada na formação de uma tabela que relaciona as referências dos identificadores estáticos dos objetos de

Estelle criadas nas ferramentas com aquelas referentes aos mesmos objetos criados no Compilador.

* Notation - Implementa os procedimentos que manipulam as referências das feramentas e do Compilador a respeito dos dados estáticos de Estelle, com o auxílio de *TabRef*.

* SStatic - Contém o procedimento que faz a leitura do arquivo da Forma Intermediária estática (FI90), estabelecendo o conteúdo das filas encadeadas dos dados estáticos de Estelle e monta as estruturas de dados, de acordo com as informações lidas.

* Contexto - Contém os procedimentos responsáveis pelo estabelecimento, destruição e substituição de contextos da especificação, manipulando os ponteiros das filas encadeadas dos dados dinâmicos de Estelle.

* Cronometro - Exporta procedimentos para estabelecer, zerar, interromper e calcular a contagem de tempo de um teste.

* TestaEstadoOperando - Usado no estudo de Lógica Temporal, implementa os procedimentos necessários para verificar se um determinado estado global operando está ocorrendo no contexto atual da especificação.

* Heurística - Apesar de não haver heurística na geração dos nós da Árvore de Alcançabilidade, é oferecido na plataforma para poder ser utilizado no desenvolvimento de outras novas ferramentas. Possui procedimento que dispara cada transição para aferir seus valores de heurística associados e depois retorna o contexto anterior.

* AlteraVariaveis - Implementa os procedimentos necessários para promover a alteração de determinada variável de uma instância de módulo, utilizando as diretivas da interface de variáveis declaradas na especificação, compondo o menu Passo a Passo que usa a diretiva $Afetar.

* FileInOut - Contém a implementação dos dados e procedimentos necessários para realizar as operações com arquivos, desempenhadas pelas feramentas, tais como : verificar se o disco está cheio, escrever uma string num arquivo, etc.

* HandleError - Responsável pela emissão de mensagens de erro para a tela, referentes a situações de incoerência ocorridas durante a execução dos testes.

O grupo dos módulos de "Interface" com o usuário

O bloco de Interface com o usuário é responsável pela exposição das informações na tela e pela leitura do teclado, estabelecendo os valores das estruturas de dados e fazendo com que os comandos do usuário sejam executados. Ele é composto pelos seguintes módulos:

* InterfaceHomemMaq - Faz a leitura de comandos e a abertura das ferramentas desenvolvidas na plataforma, assim como o processamento dos menus de opções.

* SetaProcura - Da mesma maneira que o módulo *Heuristica* foi mantido também para uma possível uitlização ou adaptação futura.

* MostraContexto - Expõe, na tela, o contexto atual da especificação Estelle, mostrando a configuração das instâncias de módulo, seus estados ativos, valor de suas variáveis, conteúdo da fila de seus IPs, entre outras coisas.

* MostraInformacoes - Apresenta informações a respeito do sistema e de um teste realizado por uma ferramenta, tais como: tempo de teste, espaço livre no disco corrente, tamanho dos arquivos de resultado e total de estados globais alcançados.

* MostraResultados - Este módulo pertence ao overlay *Resultado* e estabelece relação entre os dados da verificação com os arquivos de resultados, permitindo a visualização desses arquivos.

* LeEstadoOperando - Implementa os procedimentos necessários para leitura dos estados globais operandos, usados no Verificador.

* ModulosDOSestadosOperandos - Implementa todas as operações relacionadas com a definição e uso das instâncias de módulo associadas às declarações de estados operandos.

* LeLogicaTemporal - Outro módulo herdado do Verificador que foi mantido para facilitar uma posterior utilização ou adaptação por outras novas ferramentas.

* IndicaVerificação - Contém um procedimento que mostra uma tela para a interface com o ambiente externo da especificação Estelle e uma janela para indicar a evolução da do algoritmo.

* ChecagemDEentradas - Implementa a averiguação de erros nas linhas de comando fornecidas pelo usuário, para evitar um aborto do processamento de um teste por entradas inadequadas.

* ManipulacaoDEentradas - Implementa os procedimentos que mexem com o valor das cadeias de caracteres formadoras das linhas de comando das ferramentas, fornecidas pelo usuário.

* GerenciadorDEmenus - Implementa a fila encadeada cujos elementos são estruturas de dados que constituem a definição de um menu ou de uma janela, assim como a manipulação dos mesmos.

3.4.2.2. Os Algoritmos dos Executores da Geração

Os dois módulos que implementam os Executores da Geração são os "motores" das ferramentas desenvolvidas, sendo que todos os outros módulos da plataforma existem para servir o funcionamento deles dois. Portanto seus respectivos algoritmos serão explicados adiante em mais detalhes.

* *O algoritmo Passo a Passo* - Este algoritmo é bem simples, uma vez que o disparo das transições fica a cargo do usuário do Gerador. Os passos que compõem este algoritmo são:

Passo 1- Criar e mostrar na tela o menu de passo a passo.

Passo 2- Executar o passo 3, caso o usuário não tenha enviado o comando para finalizar a análise.

Passo 3- Conforme o comando fornecido pelo usuário, fazer a execução exclusivamente do passo 4 ou do 5 ou do 6.

Passo 4- Mostrar na tela o contexto atual da verificação e, em seguida, voltar para o 2 passo.

Passo 5- Mostrar na tela, o "menu" para fazer alteração das variáveis declaradas na especificação Estelle e, em seguida, voltar para o passo 2.

Passo 6- Obter o conjunto das transições disparáveis. Se houver a sinalização de bloqueio, retorne daqui para o passo 2. Caso contrário, execute o passo 7.

Passo 7- Expor ao usuário as transições disparáveis a fim de que ele escolha uma delas para ser executada, voltando logo em seguida ao passo 2.

* *O Algoritmo Direto* - Este algoritmo pode ser descrito pela seguinte seqüência de passos:

Passo 1- Gerar arquivo de descrição dos resultados.

Passo 2- Anular contexto atual do protocolo e, em seguida, inicializar o módulo da especificação, executando o INITIALIZE.

Passo 3- Armazenar o contexto atual como estado global raiz, gravando no arquivo de saída com extensão .GRF.

Passo 4- Criar conjunto de transições disparáveis.

Passo 5- Se a aplicação do passo 4 resultar em bloqueio, registre este resultado nos arquivos de saída e termine a geração.

Passo 6- Se a aplicação do passo 4 resultar em estado exaurido (sem transições aptas que ainda não foram disparadas) e o contexto atual for o estado global raiz, registre especificação limitada e vá para o passo 12 .

Passo 7- Se a aplicação do passo 4 resultar em estado exaurido, carregue para o contexto atual, o estado global, pai deste último estado obtido, e vá para o passo 12.

Passo 8- Escolha e execute uma transição disparável sem usar heurística.

Passo 9- Depois da execução da transição, se hover alguma instância de módulo criada sem ter a sua transição de inicialização disparada, registre este resultado nos arquivos de saída e termine o teste, pois isto não pode acontecer.

Passo 10- Faça o registro da transição disparada dentro da estrutura de dados do estado global, pai deste último estado obtido.

Passo 11- Se o contexto atual ainda não estiver na fila de estados globais gerados, coloque-o. Caso contrário, ou seja, ele já foi visitado, carregue para o contexto atual, o estado global, pai deste último estado obtido.

Passo 12- Se ainda não tiver sido registrado especificação limitada e o usuário não tiver abortado o teste, volte ao passo 4 (para fazer tudo denovo).

Passo 13- Se estiver registrado especificação limitada, verifique se o protocolo é reinicializável.

Passo 14- Registre os resultados finais nos arquivos de saída e termine o processamento.

3.4.2.3. Diferenças entre a versão UNIX e a DOS

Conforme previsto, além de possibilitar a realização de todo o processamento das ferramentas implementadas em uma mesma máquina, a Plataforma de Software deve possuir uma versão que possibilite isso também em máquinas operando com sistema operacional UNIX.

Para tal, foram necessárias adaptações nos módulos que implementam a ferramenta, o que proporcionou em algumas mudanças na arquitetura geral que descreve o funcionamento da plataforma.

Anteriormente, a estrutura da arquitetura da Plataforma de Software foi apresentada com o auxílio de uma divisão classificatória dos módulos que formam grupos fundamentais, onde os módulos atribuídos a um mesmo grupo possúem objetivos de funcionamento semelhantes.

A maioria destes grupos se manteve na arquitetura, entretanto, alguns com modificações e outros até deixaram de existir, o que poderá ser melhor percebido se for feita uma comparação entre a fig. 3.4 e a fig. 3.5.

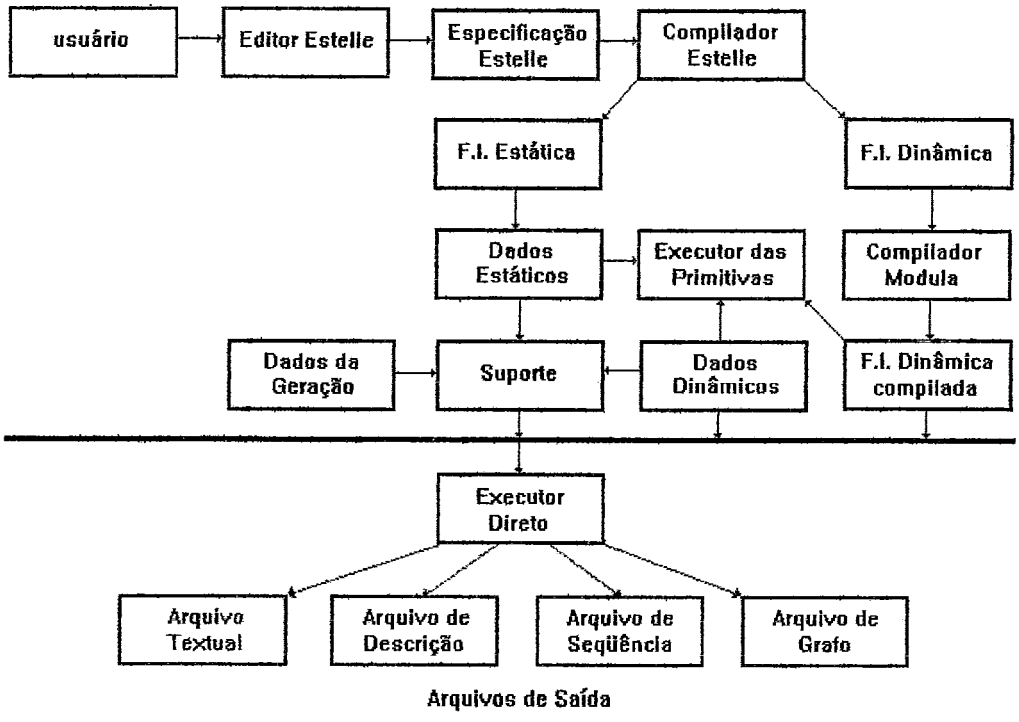

Figura 3.5 - Arquitetura da Plataforma de Software na versão para UNIX.

De início, o grupo *Intreface H/M*, observado apenas na fig. 3.4, deixou de existir devido a não necessidade de implementação de uma interface sofisticada com múltiplas janelas, conforme acontecia na versão para DOS das ferramentas. Isto acontece devido o sistema operacional UNIX ser multitarefa, e a interface do próprio SUNOS (versão de Windows da SUN), presente nas SPARCstations (estações de trabalho onde foi desenvolvida a nova versão da ferramenta), possibilitar o recurso de exposição de múltiplas janelas ao mesmo tempo, dispensando a necesidade deste serviço ser implementado na ferramenta.

Um recurso da versão DOS que não foi adaptado para a versão UNIX por não ter sido considerado fundamental, mas que pode vir a ser incluído numa outra

transição de cada vez. Este grupo é observado apenas na fig. 3.4 que descreve a uma proposta doSistema de Auxílio numa versão para PCs e compatíveis.

Como a versão para UNIX não contém a opção Passo a Passo, a interface desta versão ficou mais simples ainda e foi implementada portanto, através de comandos de linha, para ficar funcional, enquanto não se faz a integração de todas as ferramentas do Sistema de Auxílio ao Projeto de Protocolos.

Como a versão para DOS possui uma interface mais elaborada, foi feito um Manual do Usuário no Anexo A, para auxiliar na sua compreensão.

Finalmente, outra diferença é no que diz respeito aos arquivos de saída gerados pelas ferramenta desenvolvidas. Como a versão UNIX possui uma interface simplificada, esta não apresenta, durante a execução, os detalhes que descrevem cada um dos estados globais gerados obtidos, entretanto esta informação é vital para a realização dos estudos a respeito do comportamento do protocolo em teste. Portanto, na versão para UNIX, é gerado um arquivo de saída adicional, que contém de forma textual, toda essa descrição de cada um dos estados globais obtidos pelo algoritmo de busca direta, e este arquivo possui extensão .EGL (vide fig. 3.5).

Estas são, em síntese, as principais diferenças estruturais entre as duas versões oferecidas, entretanto os resultados dos testes das ferramentas implementadas com os protocolos de comunicação foram os mesmos, tanto numa quanto na outra versão do software.

3.4.3. Uma Análise da Explosão de Estados

A geração de um grafo pode ser feita de muitas maneiras. Dependendo das características particulares do grafo em questão, uma maneira pode chegar ao grafo total antes da outra.

Como todos os nós dos grafos que temos interesse em estudar são interconectados a todos os outros através de pelo menos um caminho, podemos classificar nossos grafos como árvores.

Se uma árvore é mais "larga" do que "profunda", sua obtenção completa é atingida mais rapidamente quando a geração é feita através de uma "busca em largura".

Quando ocorre o caso contrário, a geração da árvore é mais eficiente quando feita através de uma "busca em profundidade".

Como não se tem qualquer informação quanto à uma possível maior incidência do formato dos grafos que descrevem as árvore de alcançabilidade dos sistemas de comunicação em geral, ser mais largo ou mais profundo, foi adotado o método de busca em profundidade, pois se não se sabe qual é o melhor, sabe-se pelo menos que algum deles deverá ser.

Até então não foi adotada qualquer medida para reduzir o número de estados globais obtidos, nem sequer alguma estratégia para tornar a obtenção dos mesmos mais rápida, entretanto, este problema existe e as análises dos sistemas de comunicação sempre acabam esbarrando nos limites impostos pelos recursos computacionais.

Portanto devem ser estudadas maneiras de minimizar o problema, já que ele é inevitável.

3.4.3.1. A adoção de Restrições na TDF

Conforme foi visto na seção 2.2.2.1.a, a execução de uma especificação Estelle é governada pelos atributos dos módulos e a estrutura hierárquica da especificação. Esta execução pode ser de três tipos: paralela síncrona, paralela assíncrona e não determinística.

O trabalho desenvolvido em [19], mostra um estudo que conclui que as árvores geradas por sistemas com módulos que têm execução paralela, possuem árvores de alcançabilidade com maior número de nós do que teriam, se o mesmo sistema funcionasse com seus módulos executando de maneira não determinística, ou seja, com uma transição disparando de cada vez. Com base no mesmo estudo, a ferramenta desenvolvida em [19] faz as mesmas considerações.

Como medida tomada no sentido de conter a explosão de estados durante a geração da árvore, foi decidido restringir o estudo, aos casos em que os sistemas não possuem paralelismo. Para implementar esta resolução, os módulos pai da especificação, não podem ser classificados como SYSTEMPROCESS, restando apenas a classificação SYSTEMACTIVITY. Como os módulos SYSTEMACTIVITY só podem ser refinados em módulos do tipo ACTIVITY, o atributo PROCESS acabou sendo excluído também, de maneira indireta.

3.4.3.2. A possibilidade de inclusão de Heurísticas

Outra idéia, seria utilizar alguma heurística que podasse caminhos da árvore de pouco interesse, entretanto este tipo de escolha é muito delicada e exige um estudo cuidadoso para escolher tal heurística. Como a adoção desta técnica não faz parte dos objetivos deste trabalho, seguem-se algumas sugestões para orientar esta implementação no desenvolvimento das novas ferramentas para o CAD de Estelle.

Em geral, uma heurística costuma acrescentar valores calculados a cada ramo da árvore, antes que seja escolhido por qual deles o sistema deve tentar evoluir. Estes valores conferem uma espécie de classificação às transições quanto ao aspecto de qual deve ser a mais indicada de ser disparada.

Caso, posteriormente se resolva ingressar por este método para melhorar a performace da geração dos estados globais, sem comprometer a capacidade da

ferramenta de analisar os sistemas, tal heurística pode ser incluída de três maneiras na arquitetura do sistema.

Primeiro modo

Pensando numa futura utilização na implementação de novas ferramentas baseadas em algoritmos de busca para o Sistema de Auxílio, e ainda, visando facilitar a implementação de heurísticas em tais algoritmos, foram incluídos, apesar de atualmente estarem inativos, alguns módulos com este objetivo.

Existe um recurso de utilização de variáveis definidas pelo usuário, que são utilizadas no cálculo de uma heurística específica de sua implementação.

A cada transição que o usuário tenha interesse, pode ser atribuído o valor de uma heurística. Após feito isto, pode ser utilizada uma opção de gerar a árvore, disparando primeiro aquelas transições que possuem o menor valor de heurística associado, durante o funcionamento do executor do algoritmo.

Os módulos que implementam estes procedimentos poderiam ser adaptados para implementar uma outra heurística que fosse de interesse.

Anteriormente eles foram utilizados nos estudos de Lógica Temporal feitos pelo Verificador.

Segundo modo

Ainda aproveitando recursos de software do Verificador que já foram implementados, e que podem ser adaptados, existe a possibilidade de utilizar um recurso especial com o qual se pode definir todas as características da família de estados globais de pouco interesse.

Todos os estados gerados são analisados para ver se são identificadas essas características não desejáveis. Tais estados são então, podados da árvore de estados globais, ou seja, não são gravados na estrutura de dados que contém registrados todos os estados obtidos pelo Executor, após sua respectiva geração.

Esta seria outra maneira de incluir a implementação de uma heurística específica no funcionamento dos algoritmos.

Terceiro modo

A implementação de uma heurística na arquitetura de uma ferramenta, também pode ser feita sem utilizar a estratégia de aproveitar recursos de módulos remanescentes do Verificador. Esta inclusão pode ser feita diretamente no corpo dos programas que implementam a ferramenta na qual se deseja adaptar este serviço, entretanto, deste modo, a característica de todo o software ter sido desenvolvido de maneira modular e com encapsulamento de dados, seria perdida, apesar de talvez ser a maneira mais rápida e pouco elegante.

No caso de nossa ferramenta que exemplifica a aplicação da plataforma, o módulo que deveria ser alterado, seria o *Propriedades*, que implementa o funcionamento do *Executor Direto* da geração, que obtém as árvores de alcançabilidade de uma vez só, sem a interferência do usuário.

3.5. Conclusão

Para tornar claro o entendimento dos princípios básicos da implementação e da utilização da Plataforma de Software, foram descritas inicialmente, neste capítulo, as características do Sistema de Auxílio, que é o pacote de ferramentas

que englobará todas as ferramentas, e a Forma Intermediária, que é o objeto de entrada da maioria destas ferramentas.

Em seguida, foram explicados o funcionamento da Plataforma de Software, as características de sua arquitetura e seus detalhes de implementação, assim como as diferenças de implementação e do funcionamento das duas versões de todo o software que foi desenvolvido.

Também foi mostrada a utilização desta infra estrutura para a implementação de ferramentas de estudo e desenvolvimento de protocolos de comunicação, assim como a aplicação da cláusula de probabilidade.

Ainda neste capítulo, foi feita uma análise da explosão de estados globais, falando de suas implicações e sugerindo possíveis maneiras de contornar este problema nos algoritmos de busca, quando implementados nesta plataforma computacional de desenvolvimento.

Capítulo 4

4. Exemplos e Resultados

4.1 Introdução

Neste capítulo são apresentados os sistemas exemplos que foram testados com o Gerador Total de Árvores de Alcançabilidade Probabilísticas para demonstrar a utilização da Plataforma de Software para desenvolver ferramentas.

Através da comparação dos resultados obtidos com os resultados de outras ferramentas a respeito dos mesmos protocolos, foi possível validar a ferramenta exemplo desenvolvida, assim como reforçar, por analogia, a veracidade dos resultados gerados pela mesma.

São oito os sistemas testados. O problema da "Exclusão Mútua" é abordado na seção 4.2. Na seção 4.3 é apresentado o sistema "Exemplo Inverso". O protocolo "Produtor e o Consumidor" é estudado em 4.4. Na seção 4.5 é visto o "Stop and Wait" e em 4.6 é mostrado o exemplo "Sem Probabilidade". Em 4.7 é mostrado o "Abracadabra" e em 4.8 o "CasaDec". Finalmente o "SomaDif" é visto na seção 4.9. A listagem das especificações em Estelle dos protocolos testados e seus respectivos arquivos de saída gerados pela ferramenta podem ser encontrados no anexo B, no final da documentação.

4.2 O exemplo da "Exclusão Mútua"

O problema clássico da "Exclusão Mútua" foi modelado numa especificação Estelle para servir como teste inicial do Gerador de Árvores de Alcançabilidade. A estratégia de testar, inicialmente, protocolos simples que têm seus comportamentos e resultados bem conhecidos, foi adotada para que se possa

adquirir alguma confiança na ferramenta, comparando os resultados obtidos, para depois então testarmos sistemas menos explorados.

a) Uma Descrição Informal

Um sistema possui três entidades distintas. Uma delas produz um recurso que é oferecido para as outras duas entidades, entretanto este recurso só pode ser utilizado por uma das duas de cada vez. Enquanto um dos processos se utiliza do recurso produzido, o outro processo fica aguardando o recurso ser liberado para que só então, também possa desfrutar do mesmo.

b) Uma Descrição em Máquinas de Estado

No modelo em máquinas de estado descritas em Estelle a especificação ficou com um módulo principal representando o sistema como um todo e é do tipo System Activity. Este possui três instâncias como módulos filhos. Uma delas representa o gerador do Recurso e é do tipo Activity. As outras duas são os Processos que concorrem pelo recurso e ambos são do tipo Activity também. Estes dois últimos possuem comportamento idêntico, portanto são instâncias de um mesmo módulo. Esta arquitetura é mostrada na fig. 4.1.

O comportamento dos módulos filhos será descrito a seguir em máquinas de estado pelas figuras 4.1 e 4.2 para tornar bem claro o funcionamento destas instâncias e exemplificar o tipo de aspecto que é abordado durante uma especificação em Estelle.

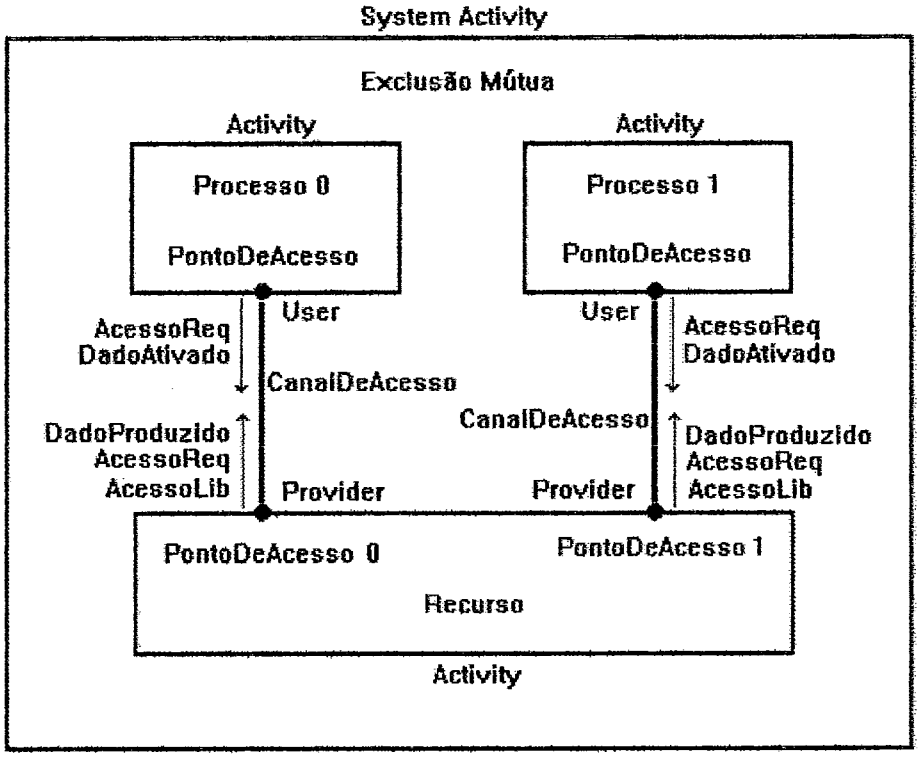

Figura 4.1 - Estrutura da especificação Estelle do protocolo "Exclusão Mútua".

Os Processos podem enviar dois tipos de mensagens: AcessoReq e DadoAtivado. AcessoReq é um pedido para utilizar o recurso. DadoAtivado é uma indicação de que o recurso já foi utilizado e não se faz mais necessário, por enquanto.

O Recurso pode enviar três tipos de mensagens: DadoProduzido, AcessoNeg e AcessoLib. DadoProduzido é o próprio recurso tão disputado pelos Processos. AcessoNeg é a indicação de que o recurso já esta sendo utilizado pelo outro Processo. AcessoLib é a indicação de que o recurso já está disponível.

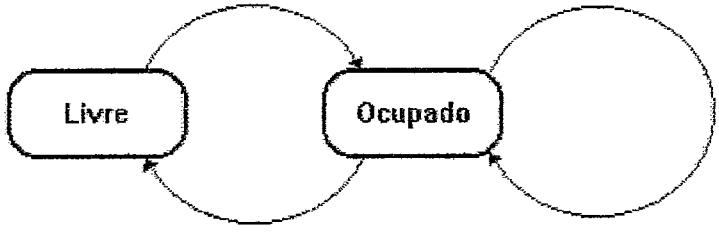

Figura 4.2 - Máquina de Estados de uma instância do tipo Recurso.

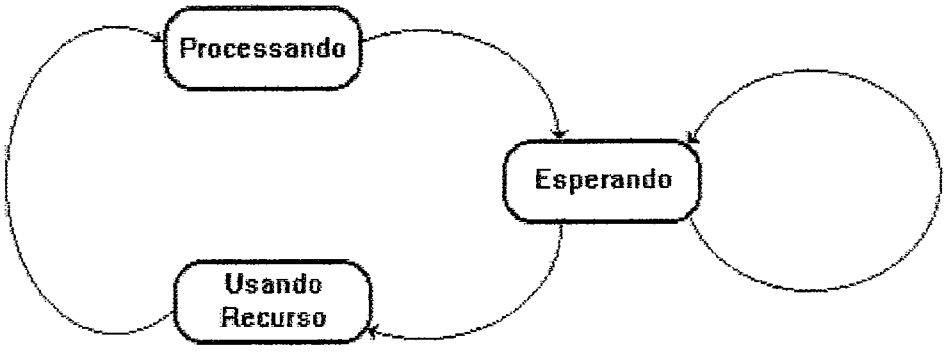

Figura 4.3 - Máquina de Estados de uma instância do tipo Processo.

c) Resultados

Este mesmo protocolo foi testado anteriormente pelo Verificador para Estelle, outra das ferramentas que compoem o Sistema de Auxílio, e por isso o problema da "Exclusão Mútua" foi escolhido para servir como um dos testes de averiguação do funcionamento do algoritmo implementado na Plataforma de Software.

A ferramenta concluiu que o sistema especificado pode alcançar um total de 31 Estados Globais diferentes, tendo percorrido toda a árvore de alcançabilidade. A ferramenta ainda classificou o protocolo como sendo

Limitado, por possuir uma quantidade finita de Estados Globais alcançáveis, e Reinicializável, por ter a característica de sempre poder retornar ao estado inicial raiz.

Os resultados obtidos foram totalmente coerentes com os da outra ferramenta, o que nos faz crer que as respostas obtidas estão corretas. Entretanto existe a possibilidade, ainda que pequena, das duas ferramentas estarem cometendo o mesmo erro. Portanto seria aconselhável acrescentar, no conjunto de protocolos testados, um que tivesse sua Cadeia de Markov de tempo discreto conhecida para sanar definitivamente essas dúvidas, já que uma árvore de 31 estados não é pouco trabalhosa de calcular manualmente.

4.3 O sistema "Exemplo Inverso"

O propósito deste sistema é de testar a ferramenta em desenvolvimento com um protocolo que possua sua C.M. (Cadeia de Markov) de tempo discreto conhecida, portanto ele foi concebido de maneira inversa, ou seja, a partir da C.M. até se chegar na sua respectiva especificação formal.

Uma árvore de alcançabilidade conhecida de ante mão com a característica de, a partir de um mesmo estado, poder evoluir para vários outros, é muito conveniente, pois com esta topologia, que será mostrada adiante, o algoritmo de busca vai poder ser bem testado.

a) Uma Descrição Informal

Esta especificação não é a representação de nenhum sistema de comunicação real conhecido, e como foi concebido de maneira inversa à maneira usual, que é, a partir da especificação, obter-se a árvore de estados globais, ele foi

denominado simplesmente como "Exemplo Inverso", pois foi criado apenas com a função de ser testado pela ferramenta e ajudar no seu desenvolvimento.

b) Uma Descrição em Máquinas de Estado

Para tornar mais fácil o trabalho de fazer uma especificação Estelle concebida a partir da árvore de estados globais que a mesma deverá gerar, após passar pela ferramenta, foi idealizado um sistema bem simples, composto por um módulo único instanciado apenas uma vez. Portanto não podem existir canais de comunicação externos.

Poderia haver IPs internos, mas não se considerou importante associar a mudança de estado à transmissão e ao recebimento de mensagens e além disto este tipo de associação já havia sido testada no exemplo anterior.

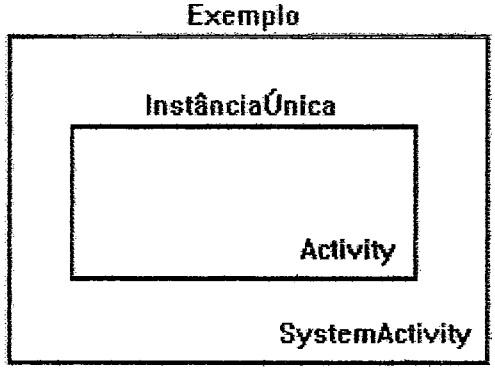

Figura 4.4 - Estrutura da especificação Estelle do sistema "Exemplo Inverso".

c) Resultados

Como os resultados obtidos pela ferramenta foram os mesmos que os previstos, já que a especificação foi feita a partir de uma C.M. particular, o teste mostrou que o algoritmo de geração da árvore de alcançabilidade tem um funcionamento confiável, esvanecendo as dúvidas que ainda existiam após o primeiro teste.

O Gerador obteve 9 Estados Globais e classificou o sistema como Limitado e Reinicializável, conforme podemos conferir na fig. 4.5.

Este mesmo sistema ainda foi testado pelas ferramentas descritas em [16] e [19], tendo seus resultados idênticos com os da ferramenta exemplo.

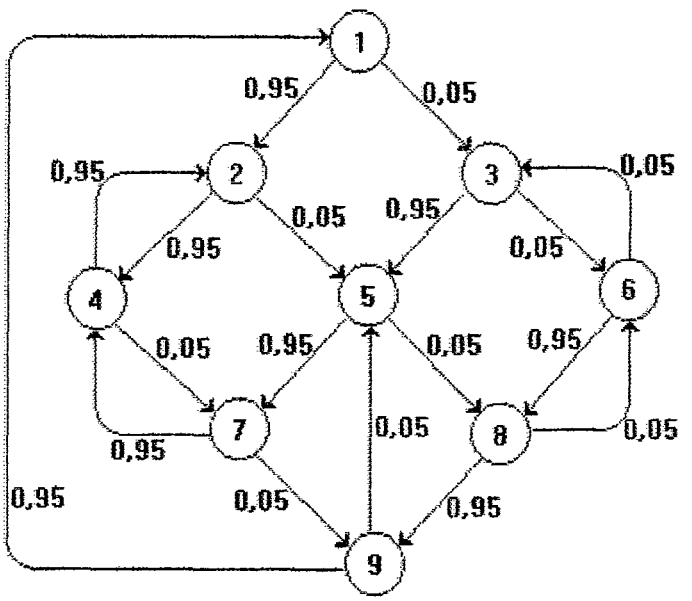

Figura 4.5 - C.M. de tempo discreto do sistema "Exemplo Inverso".

4.4 O exemplo do "Produtor e o Consumidor"

Este é outro exemplo clássico no estudo de protocolos de comunicação que, apesar de simples, é muito útil no sentido de averiguar a confiabilidade dos resultados obtidos pela ferramenta pois estes resultados são conhecidos e podem ser conferidos manualmente por ser um sistema que gera alguns poucos Estados Globais.

a) Uma Descrição Informal

Considere um sistema formado por dois processos, um deles Produtor de alguma informação e o outro o Consumidor da informação produzida. O processo Produtor só envia a informação produzida quando o processo Consumidor está pronto para consumir a informação gerada. caso contrário ele nada faz.

b) Uma Descrição em Máquinas de Estado

O modelo em máquinas de estado que foi descrito em Estelle ficou composto por um módulo System Activity contendo dois módulos filhos Activity interconectados por um canal por onde são transmitidas e recebidas as mensagens.

A mensagem produzida pelo processo Produtor é denominada MSG, ao passo que a indicação de que o processo Consumidor já está apto a consumir mensagens é denominada de LIB.

Nesta especificação não foram definidos diferentes Estados Locais para cada uma das instâncias de módulo. Estas foram descritas como se cada uma só possuísse um único Estado Local que possui diversos comportamentos. Isto é

apenas uma questão de ponto de vista, pois seria o mesmo que subdividir estes diferentes comportamentos em vários estados.

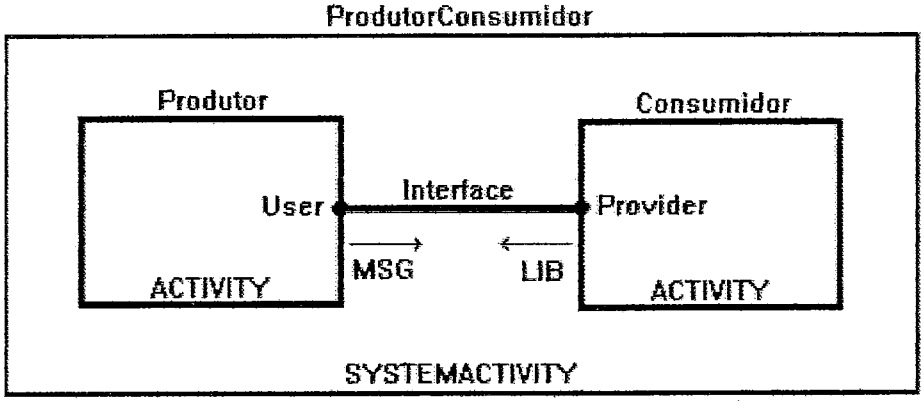

Figura 4.6 - Estrutura da especificação Estelle do protocolo "Produtor e o Consumidor".

c) Resultados

A ferramenta exemplo obteve um total de três Estados Globais alcançados tendo esgotado toda a árvore de alcançabilidade do sistema especificado. Classificou o sistema especificado como sendo Limitado, pois possui um número finito de estados, e não reinicializável, devido a impossibilidade de se retornar ao Estado Raíz (estado 1).

A árvore de alcançabilidade obtida é mostrada acima e todos estes resultados estão de acordo com os obtidos pela ferramenta de avaliação [19] e pelo Verificador descrito em [16].

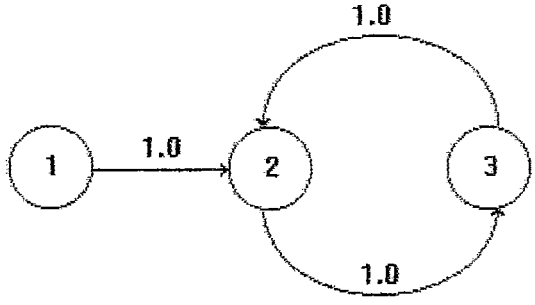

Figura 4.7 - Árvore de Estados do protocolo "Produtor e o Consumidor".

4.5 O exemplo "Stop and Wait"

O sistema especificado aqui é uma versão do protocolo "Stop and Wait" e foi adotado desta maneira por já ter sido assim testado pela ferramenta em [19].

a) Uma Descrição em Máquinas de Estado

A especificação deste protocolo é formada por quatro módulos: Clock, Transmissor, Receptor e Rede.

O módulo Clock é encarregado de controlar o tempo de retransmissão de mensagens. Quando uma mensagem vai ser enviada, o módulo Transmissor inicia o Clock com a mensagem StartTimer. O tempo de espera pelo Ack tem distribuição geométrica com média Del.

O módulo Transmissor sempre tem mensagens para transmitir. Quando ele se encontra no estado Estab, tenta transmitir iniciando o Clock e entrando num estado de processamento de informação, para em seguida transmitir a mensagem com um dado número de seqüência.

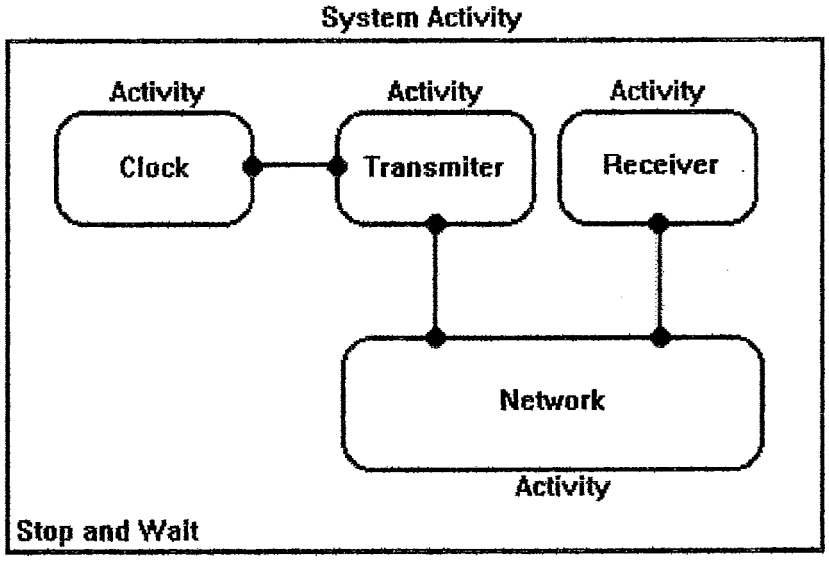

Figura 4.8 - Estrutura da especificação Estelle do protocolo "Stop and Wait".

O módulo Receptor, ao receber uma mensagem, a processa e envia o Ack ao módulo Transmissor, caso esta mensagem não tenha sido corrompida na Rede.

O módulo de Rede especifica uma rede de transmissão que pode perder mensagens.

b) Resultados

O interesse de testar este protocolo residia no fato dele já ter sido testado anteriormente pela ferramenta descrita em [19] e terem sido detectados bloqueios durante sua execução. Ainda não havia sido testado, na ferramenta, nenhum protocolo de comunicação com tais características.

Confirmando o resultado obtido pela outra ferramenta, o programa concluiu que a especificação desta versão do protocolo "Stop and Wait" realmente possui bloqueio, que foi logo identificado.

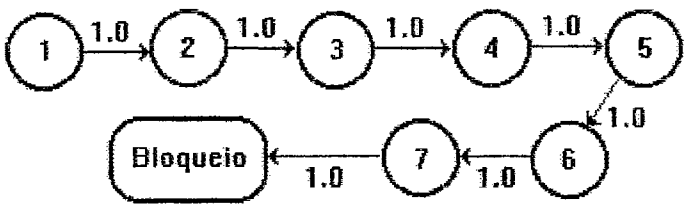

Figura 4.9 - Árvore de alcançabilidade parcial do protocolo "Stop and Wait".

O teste obteve menos de dez Estados Globais, conforme mostra a fig. 4.9, pois sempre que a ferramenta detecta a ocorrência de um bloqueio, ela interrompe o teste e informa ao usuário, dizendo que este foi o motivo da finalização do teste.

4.6 O exemplo "Sem Probabilidade"

O objetivo deste exemplo é mostrar ao usuário, como a ferramenta se comporta quando introduzimos para análise, um protocolo com erro de especificação. No caso, o erro é a falta da descrição da probabilidade de disparo de qualquer das transições da especificação. Como já foi dito, todas as especificações necessitam da declaração do valor da probabilidade de disparo de todas as transições. Para testar esta exigência, foi desenvolvido um serviço especial que será apresentado com este teste.

a) A Descrição do Protocolo

A descrição informal deste protocolo, assim como a descrição em máquinas de estado, são as mesmas das seções 4.4.a e 4.4.b, respectivamente, já que este teste é uma experiência de forçar um erro na especificação do protocolo "Produtor e o Consumidor" para verificarmos o comportamento da ferramenta.

b) Resultados

Durante o teste desta especificação, a ferramenta interrompe a execução e abre uma janela de aviso indicando que houve um erro. Informa ao usuário que encontrou uma transição que não contém a descrição da probabilidade de disparo e ainda indica em qual transição de qual módulo isto ocorre, para facilitar a localização e correção da falha na especificação por parte do usuário.

A tarefa de ajustes da especificação se torna mais fácil ainda quando o usuário utiliza a cláusula NAME, oferecida pela própria Linguagem de Descrição, para identificar cada uma das transições. Foi pensando neste recurso, que este serviço de proteção contra especificações mal elaboradas foi desenvolvido.

Este serviço foi implementado para suprir o fato de o Compilador Estelle [14] não fazer este tipo de averiguação, já que a cláusula PROBABILITY não faz parte do conjunto de palavras reservadas do Estelle padrão.

Como outras ferramentas do Sistema de Auxílio não precisam fazer a exigência desta declaração, foi considerado mais prudente fazer este tipo de compilação exclusivamente durante o processamento da ferramenta exemplo, ao invés de alterar o código do próprio Compilador, pois isto implicaria na geração de erro quando o compilador fosse usado numa especificação em que não se

pretende fazê-la passar por esta ferramenta, e por sua vez, não tem que possuir a cláusula PROBABILITY declarada obrigatoriamente.

4.7 O exemplo "Abracadabra"

A idéia de testar este protocolo surgiu com a necessidade de se fazer uma experiência de executar o programa por um longo tempo, para verificar o seu comportamento, se não ocorreria nenhuma falha de funcionamento e observar sua performace trabalhando um protocolo de muitos estados globais.

Para tal, deixou-se testar o protocolo Abracadabra, que conhecidamente possui um número muito grande de possíveis estados globais, desde 14:00 até 23:30 aproximadamente, num equipamento 486 DX2 66MHz.

a) A Descrição do Protocolo

O protocolo ABRACADABRA foi padronizado em conjunto pela ISO e pelo CCITT para ser usado exclusivamente em testes. Ele é u 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).

Nesta versão, o corpo da entidade ***usuário*** foi implementado de forma simplificada, tendo apenas o objetivo de fornecer à camada ***abracadabra***, as primitivas de requisição e resposta de serviço para promover o seu teste.

Portanto os módulos com comportamento descrito por este tipo foram implementados possuindo um único estado local.

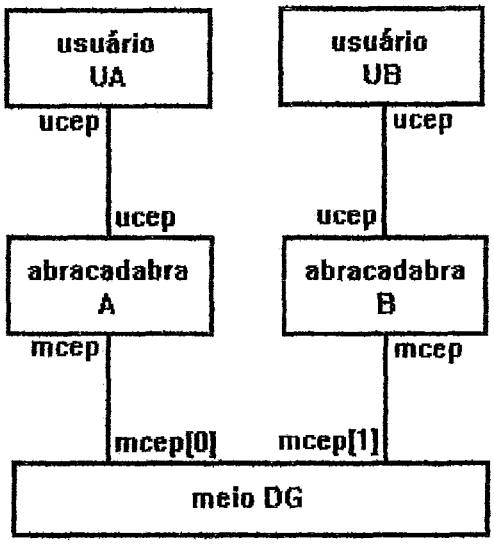

Figura 4.10 - Estrutura da especificação Estelle do protocolo "Abracadabra"

b) Resultados

Inicialmente, o Gerador detectou, em pouco mais de dois minutos, bloqueio na primeira especificação do protocolo, a qual se pensava estar correta por já ter sido testada por outras ferramentas anteriormente, e 60 estados globais foram alcançados. Depois disso, o erro de especificação pôde ser mapeado através dos recursos da ferramenta e a camada "usuário" da especificação foi ajustada.

Após um novo período de aproximadamente nove horas e meia, a ferramenta desenvolvida obteve um total de estados globais em torno de 550, quando o teste foi interrompido via usuário, por já estar funcionando por um longo tempo sem apresentar qualquer problema e ter sido considerado satisfatório. Portanto esta especificação foi classificada como "não limitada".

Devido às características do Gerador, este mesmo teste pode continuar a ser feito a partir dos arquivos de saída obtidos.

4.8 O exemplo "CasaDec"

O objetivo deste exemplo é testar o comportamento da ferramenta desenvolvida quando valores de probabilidade com mais de cinco algarismos decimais forem atribuídos.

Recomenda-se que os valores das probabilidades, associadas a cada uma das transições, possuam até cinco algarismos na parte decimal.

a) A Descrição do Protocolo

Este exemplo é uma versão do sistema "Exemplo", já testado, só que desta vez, com os valores das constantes que definem as probabilidades alterados para novos valores com mais de cinco casas decimais.

```
CONST    low = 0.049995,
         high = 0.950000;
```

b) Resultados

O que se observou foi que as constantes que possuem mais de cinco algarismos na parte decimal ficam com seus valores arredondados, conforme é mostrado a seguir:

low = 0.05000 & high = 0.95000

4.9 O exemplo "SomaDif"

Como uma especificação de um protocolo de comunicação costuma possuir um número muito grande de transições, é fácil que se esqueça de atribuir o valor da probabilidade associada a uma destas muitas transições.

Para facilitar a vida do usuário, foi implementado mais um recurso que agora está disponível na plataforma de software. Este recurso verifica se a soma dos valores das probabilidades associadas às transições que são disparadas a partir de um mesmo estado global, é diferente de 1 (um).

Este exemplo possui características propícias para validarmos ou não o bom funcionamento do novo recurso.

a) A Descrição do Protocolo

Este exemplo é uma versão do sistema "Exemplo", já apresentado anteriormente, entretanto desta vez, com alguns estados globais onde a soma das probabilidades de transição para um outro estado global é diferente de um. Com isso pode-se verificar o funcionamento e a praticidade deste recurso que auxilia na correção da especificação do protocolo de comunicação.

b) Resultados

O resultado obtido foi satisfatório, pois os estados em que a soma das probabilidades das transições de saída foram devidamente identificados e registrados, como pode ser observado no arquivo de seqüência na seção B.7.a do anexo B.

4.10 Conclusão

Todos estes testes foram fundamentais na fase de depuração dos erros de implementação da ferramenta. Concluída esta fase, o desempenho da ferramenta exemplo foi considerado satisfatório, pois foram comparados os resultados obtidos de três ferramentas distintas e todos os testes se mostraram totalmente coerentes entre si.

Além deste algoritmo de busca ter sido implementado para gerar toda a árvore de alcançabilidade de uma vez só, também foi implementada a mesma ferramenta exemplo, só que com uma filosofia de exploração de estados globais diferente, determinada por um outro algoritmo. Nesta outra versão da ferramenta, o novo algoritmo também vai obtendo os estados globais de interesse com a informação da probabilidade de transição de um para outro, só que um de cada vez, ao gosto do usuário.

A experiência mostrou que, além das ferramentas desenvolvidas serem interessantes para análise do comportamento de protocolos de comunicação, a utilidade da Plataforma de Software é muito grande neste tipo de tarefa.

Com uma plataforma computacional com tais características especiais, fica facilitada a tarefa de desenvolver ferramentas basedas em algoritmos de busca e inclusive de utilização na área de análise de desempenho.

Todo o processamento dos testes da ferramenta pode ser realizado em uma única máquina, o que já é considerado um avanço, e em tempo hábil considerado bastante satisfatório (alguns demoraram minutos e outros apenas segundos) para os protocolos testados. Estes tempos podem ser verificados nas listagens dos arquivos de saída de cada um dos protocolos, que se encontram no anexo B, no final da documentação.

Outra coisa que se pôde observar ao longo deste capítulo foi o quanto subjetivas e, algumas vezes, pouca claras, são as descrições de um protocolo de comunicação, quando feitas de maneira informal. Daí se conclui que com a popularização das Técnicas de Descrição Formal, haverá uma tendência natural de as pessoas fazerem uma opção pelo abandono das descrições informais, ou pelo menos que sempre venham acompanhadas de sua respectiva especificação formal, para que não deixe dúvidas nem margem a interpretações dúbias.

Capítulo 5

5. Conclusões

5.1 Preliminares

O objetivo deste trabalho foi implementar uma plataforma de software que já oferece de maneira organizada e encapsulada as informações provenientes das especificações em Estelle e ainda alguns recursos de manipulação destas estruturas de dados. Com isso o desenvolvimento de ferramentas que auxiliam no estudo e implementação de protocolos de comunicação ficou uma tarefa mais fácil, e padronizada.

Para que estas ferramentas sejam implementadas basta introduzir um algoritmo de busca conveniente e fazer os devidos ajustes nas interfaces com o usuário. Como a tradução dos dados de Estelle já está pronta, e todos os algoritmos irão manipular basicamente os mesmos dados, todas as ferramentas já são concebidas com a característica de possuir a Forma Intermediária gerada pelo compilador Estelle como principal entrada de dados, uma vez que as especificações de protocolos são feitas segundo as regras da TDF Estelle.

Através da utilização desta base computacional, as futuras ferramentas que deverão compor o Sistema de Auxílio ao Projeto de Protocolos de Comunicação já nascerão naturalmente neste CAD de Estelle, evitando trabalhos futuros de integração das ferramentas.

Um dos aspectos desejados era a condição de que toda ferramenta deveria ter uma versão que funcionasse totalmente em uma mesma máquina, para facilitar o seu aprendizado e utilização. Desta maneira o Sistema de Auxílio pode ser difundido mais rapidamente nos meios acadêmicos.

Estes objetivos foram alcançados, uma vez que foram feitas não só uma, mas duas versões da plataforma de software para implementação de algoritmos de

busca, com as características exigidas. Uma versão para auxiliar no desenvolvimento de ferramentas que devem funcionar completamente em DOS para servir de uso acadêmico para alunos da graduação e outra que necessita unicamente do UNIX.

Apesar de, inicialmente, esperar-se que com a versão para DOS resultassem ferramentas com uma performace inferior àquelas obtidas com o auxílio da versão para UNIX, os testes mostraram resultados contraditórios a esta espectativa. O tempo de obtenção das árvores de alcançabilidade probabilísticas foi inferior nos equipamentos compatíveis da linha PC, sendo que todos os outros resultados obtidos pelas duas versões da ferramenta exemplo implementada foram idênticos.

Para que todo o software sempre tenha total compatibilidade de funcionamento com qualquer equipamento da família dos PCs, foi necessário fazer a implementação desta plataforma computacional, utilizando-se do recurso de subdivisão do processamento em overlays. A adoção deste procedimento foi imperativa devido os modelos mais antigos da linha PC possuírem pouca quantidade de memória RAM, o que impede que seja feito o carregamento de todo o programa na memória durante sua execução, problema que já não existe nas estações de trabalho SPARC. Entretanto os modelos mais atuais compatíveis com a linha PC não sofrem desta escassez de memória randômica. Todos os testes da versão para DOS foram realizados em equipamentos com processadores 486, trabalhando com freqüências de operação da mesma ordem das freqüências das estações de trabalho que funcionam sob UNIX.

Estes tempos apesar de muito semelhantes, foram maiores quando obtidos a partir da versão para UNIX. Por exemplo, o sistema denominado Exemplo Inverso, foi calculado em 3 segundos por um 486 DX2 66MHz, em 15 segundos por um 486 DX 40MHz e em 19 segundos por uma SPARCstation.

Os vários testes feitos com diferentes sistemas estudados anteriormente em outros trabalhos validaram o bom funcionamento e a veracidade dos resultados gerados pela ferramenta exemplo implementada.

5.2 Benefícios

Inicialmente, com a conclusão deste trabalho, foi padronizado o desenvolvimento de novas ferramentas para o Sistema de Auxílio ao Projeto de Protocolos de Comunicação para Redes de Computadores que irá tornar a tarefa de desenvolvimento e depuração de protocolos um trabalho mais organizado, menos sujeito à falhas, menos cansativo e mais rápido.

O desenvolvimento desta Plataforma de Software também determinou uma melhora no aspecto da utilização dos softwares que se seguirão em relação ao aspecto da computação ser feita toda ela em um único equipamento, o que nem sempre aconteceu no passado, o que era pouco prático. Portanto esta plataforma de software definiu uma nova política de desenvolvimento de ferramentas baseadas em algoritmos de busca substituindo a maneira como era usual de desenvolver tais ferramentas para o Sistema de Auxílio.

A disponibilidade das ferramentas desenvolvidas para exemplificar a aplicação da plataforma de software, com suas características especiais, permite ao projetista vislumbrar todos os possíveis estados globais alcançáveis pelo seu protocolo e navegar pelos mesmos para depurar erros e prever comportamentos de forma mais precisa e eficiente, o que antes não era possível.

Finalmente, pode-se citar ainda a otimização dos investimentos em desenvolvimento, a melhoria da qualidade dos sistemas de comunicação gerados e o aumento da segurança e confiabilidade dos protocolos projetados como mais alguns benefícios que o desenvolvimento desta ferramenta acarretou.

5.3 Perspectivas

Uma das aplicações que já pode começar a ser feita é a utilização desta plataforma para fazer a integração das ferramentas já desenvolvidas do Sistema de Auxílio ao Projeto de Protocolos de Comunicação para Redes de Computadores que, necessita possuir uma interface sofisticada e amigável para todos os seus recursos, o que trará mais conforto ao usuário.

Um trabalho de integração das ferramentas já existentes, já está sendo feito, entretanto sem utilizar uma plataforma computacional única. Este novo sistema irá funcionar totalmente em sistema operacional UNIX, tornando sua utilização eficiente e aumentando a possibilidade de divulgação e intercâmbio do mesmo com outros centros de desenvolvimento, que mais freqüêntemente costumam desenvolver ferramentas para este mesmo sistema operacional, entretanto, tal sistema conterá muito código replicado desnecessariamente, ocupando muito espaço em memória.

Um aspecto que ainda pode ser explorado é no que diz respeito à geração do grafo que descreve a árvore de alcançabilidade de estados globais. O processamento de dados pode vir a ser tão intenso que, mesmo as estações de trabalho SPARC da SUN, e os atuais 486s e Pentiuns podem vir a ficar lentos nesta tarefa.

Uma alternativa para minimizar o problema e obter resultados de maneira mais rápida seria a aplicação de heurísticas convenientes para podar caminhos da árvore de estados que levassem a estados globais de pouco interesse, já que a geração desta árvore está sendo completa e sem heurísticas.

Outra alternativa para maximizar o desempenho desta geração, é tentar explorar os recursos de programação paralela, oferecidos pelo sistema operacional

UNIX, e fazer uma reestruturação da maneira de como é feito o processamento da atual computação.

Como esta plataforma tem a preocupação de aumentar a capacidade de descrição das características dos protocolos, descritos em Estelle, através da inclusão da possibilidade de especificação de uma probabilidade associada ao disparo de cada transição, o próximo passo deverá ser o aproveitamento destes recursos da plataforma de software, para implementar uma ferramenta que obtenha Cadeias de Markov gerais de protocolos especificados com Estelle.

Uma das ferramentas exemplo implementadas mostrou que as características dos dados oferecidos pela plataforma de software já são suficientes para o desenvolvimento de ferramentas com esta finalidade, uma vez que se pode fazer uma analogia das árvores de alcançabilidade obtidas com Cadeias de Markov de tempo discreto com tempo médio de disparo de transições de uma unidade.

Uma Cadeia de Markov pode ser descrita através de uma Matriz de taxas de transição de estados, a qual permite cálculos de variáveis de interesse a análises de desempenho. Uma ferramenta com tal saída ofereceria dados preciosos para serem utilizados por outras ferramentas que implementam a automatização dos cálculos das variáveis de análise de desempenho. Existem trabalhos desenvolvidos que abordam esta questão e a integração de mais este recurso no Sistema de Auxílio seria de grande valia.

Bibliografia

Bibliografia

[01] - Barbacci , M. R. ; "A Comparison of Register Transfer Language for Describing Computers and Digital Systems" ; IEEE - Transactions on Computers ; Vol. C-24 ; Nº 2 ; Feb. 1975.

[02] - VanCleemput, W. M. ; "Computer Hardware Description Languages and their Applications" ; 16th Design Automation Conference Proceedings ; SanDiego - California ; June 1979.

[03] - ISO/TC 97/SC 21 ; "Information Processing Systems - Open Interconnection - Estelle - A Formal Description Technique Based on an Extended State Transition Model" ; Output of Editing Meeting ; Paris ; June 1988.

[04] - Budkowski, S. ; Dembinski, P. ; "An Introduction to Estelle: A Specification Language for Distributed Systems" ; Computer Networks and ISDN Systems ; Nº 14 ; 1987.

[05] - Souza, W. L.; "Estelle: Uma Técnica para a Descrição Formal de Serviços e Protocolos de Comunicação" ; Revista Brasileira de Computação ; Vol.5; Jul./Set. 1989.

[06] - Courtiat, J. P. ; Dembinski, P. ; Groz, R. ; Jard, C. ; "Estelle: Un Language ISO Pour Les Algorithmes Distribués et Les Protocoles" ; Rapports de Recherche, Inria Rennes ; Nº 595 ; Décember 1986.

[07] - Sanches, E. G. ; "Especificação de Protocolo de Comunicação Via Rádio em Estelle" ; Projeto Final de Graduação ; Departamento de Eletrônica ; Escola de Engenharia ; UFRJ ; 1991.

[08] - Wirth, N., "Programmind in MODULA-2", Springer-Verlag, 1988.

[09] - Ford, G. e Wiener, R., "MODULA-2 A Software Development Approach", Jonn Wiley & Sons, Colorado Springs, 1985.

[10] - Tanenbaum, A. S., "Computer Networks", Prentice Hall, second edition, 1988.

[11] - Pedroza, A. C. P. ; Goulart, C. C. ; Valim, P. R. ; Olveira Jr., R. C. de ; "Um Sistema de Auxílio ao Projeto de Protocolos de Comunicação para Redes de Computadores" ; Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos ; Florianópolis ; de 11 a 14 Set. 1989.

[12] - Silva, G. S. e Pedroza, A. C. P., "Geração Automática de Seqüências de Testes para Protocolos de Comunicação: Uma Aplicação em Sinalização Telefônica", 1º Simpósio de Automação Integrada, 1990.

[13] - Valim, P. R. O., "Um Simulador de Protocolos de Comunicação para a Linguagem ESTELLE", Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1990.

[14] - Oliveira Jr., R. C., "Um Compilador para a Linguagem ESTELLE", Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1991.

[15] - Goulart, C. C., "Um Sistema de Edição Dedicado para a Linguagem ESTELLE, Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1991.

[16] - Filho, O. B., "Verificador para ESTELLE", Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1992.

[17] - Berson, S., Silva, E. S. and Muntz, R. R., "An Object Oriented Methodology for the Specification of Markov Models", First International Conference on Numerical Solution of Markov Chains, 1990.

[18] - Diniz, M. C., "Ferramenta para Especificação e Geração de Modelos", Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1992.

[19] - Ochoa, P. E. M., "Avaliação de Desempenho de Sistemas Distribuídos Especificados em ESTELLE", Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1990.

[20] - Rodrigues, R. L., "Síntese de Alto Nível de Protocolos de Redes Especificadas na Linguagem Estelle", Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1994.

[21] - Alcântara, S. C., "Implementação Semi-Automática de Protocolos Descritos em Estelle", Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1993.

[22] - Courtiat, J. P., "ESTELLE* - A Powerful Dialect of Estelle for OSI

Protocol Description", Technical Report, SEDOS, 1988.

[23] - Ross, S. M., "Stochastic Processes", John Wiley and Sons, California, 1983.

[24] - Melamed, B., et al., "Randomization procedures in the computation of cumulative time distributions over discrete state Markov processes", Operations Research, August 1984.

[25] - Gross, D., et al., "The randomization technique as a modeling tool and solution procedure for transient Markov processes", Operations Research, April 1984.

[26] - Kleinrock, L., "Queueing Systems, Volume II, Computer Applications", Wiley, N.Y., 1976.

[27] - Bertsekas, D.& Gallagar, R., "Data Networks", Prentice Hall, 1987.

[28] - Bochmann, G. V., et alli, "Implementation Support Tools for OSI Application Layer Protocols", Technical Report nº 748, Université de Montréal, 1990.

Anexo A

ANEXO A - Manual do Usuário

A.1 - Organização Estrutural da Plataforma de Software

A versão para DOS da Plataforma de Software possui uma estrutura organizada segundo a técnica de overlay, pois o total do seu código de implementação ficou muito grande, não sendo possível residir no espaço de memória de 640 Kbytes dos computadores pessoais compatíveis com os da linha IBM, para os quais a sua versão inicial se destina., o que já não acontece com a versão para UNIX por não existir esta dificuldade. Para a desenvolvimento de um algoritmo de busca qualquer, aplicado à Plataforma de Software, tal algoritmo deve ser inserido em um overlay próprio, no caso de utilização da versão PC.

Além da camada base (arquivo com extensão .EXE), existem mais seis "overlays" (arquivos com extensão .OVL) que compõem o sistema. Logo abaixo é feita a descrição destes "overlays" por meio dos seus módulos principais.

* AvalEst - Este é o módulo principal da camada base responsável pela execução dos Algoritmos de Busca. A camada base é composta de todos os módulos referentes às estruturas de dados e do módulo leitor de comandos de abertura da interface com o ambiente externo, chamado de InterfaceHomemMaq.

* Iniciador - Módulo principal do "overlay" responsável pela inicialização (estabelecimento ou criação) de várias estruturas de dados das ferramentas, como por exemplo: os dados referentes à Estelle (estáticos e dinâmicos); os números para as diretivas de consulta dos valores das expressões usadas na procura; e etc.

* Resultado - O "overlay" que cria e monta o menu de resultados, tem este módulo como o seu principal. As opções deste "menu" estão relacionadas à

exposição do conteúdo dos arquivos de saídas e ao estabelecimento do valor de alguns parâmetros que afetam a criação destes arquivos.

* Informações - O "overlay" que possui este módulo principal, tem como tarefa, a exposição na tela, da janela de informações. A descrição destas informações encontra-se junto ao comentário do módulo MostraInformacoes, sendo o responsável pela criação desta janela e fazendo parte também deste "overlay".

* PassoApasso e Propriedades - Módulos dos "overlays" executores dos dois algoritmos de busca, já comentados anteriormente.

* Aborto - Este é o módulo principal do "overlay" que executa o tratamento de uma interrupção (aborto) provocada pelo usuário durante qualquer tipo de execução (exceto a execução passo a passo). Quando o usuário interrompe a geração da árvore de alcançabilidade, este "overlay" é executado, mostrando um "menu", cujas opções incluem: o cancelamento definitivo da geração em curso, o retorno normal a esta geração e a apresentação das informações e dos resultados até o momento.

A.2 - Comandos da Interface

Como a versão UNIX do sistema não possui opções devido o Executor Passo a Passo não ter sido incluído nesta versão, sua interface com o usuário ficou mais simples, e sem necessidade de um menu de opções para dar início à geração da árvore de alcançabilidade.

Além disto, devido às capacidades multitarefas do sistema operacional das estações de trabalho SUN, a interface do programa com o usuário, toda elaborada e cheia de janelas da versão DOS, se torna desnecessária na versão UNIX, se soubermos explorar bem os recursos da máquina.

Uma vez sendo multitarefa, a interface na SUN pode ser bem simplificada pois a exposição dos arquivos de saída com os resultados dos eventuais testes não precisa ser controlada pela interface do sistema, como acontece na versão DOS. Na SUN, os arquivos de saída que forem de interesse podem ser "abertos" cada um em uma janela e serem observados em tempo real, enquanto em outra janela é executado o programa principal.

Portanto a interface da versão DOS se fez necessária de ser elaborada com tantos recursos, ao passo que, para a versão que funciona nas SUNs, foi feita uma interface simplificada com comandos de linha que formam um diálogo com o usuário, apenas para torná-la apta de utilização nas SPARC Stations da SUN, enquanto não é concluída esta versão do pacote de ferramentas denominado de Sistema de Auxílio a Projetos de Protocolos de Comunicação, que conterá várias outras ferramentas e possuirá uma interface sofisticada.

Desta forma, serão apresentados a seguir os comandos de utilização referentes à versão para DOS do sistema, que já é mais complexa, pois o "diálogo" da versão UNIX é bem simples.

Figura A.1 - A tela de apresentação da Plataforma de Software.

A entrada dos comandos da interface é toda realizada por meio de "menus" e janelas. No que diz respeito aos "menus", para escolher uma de suas opções, basta posicionar a barra de seleção sobre a opção desejada e em seguida pressionar a tecla ENTER ou pressionar a tecla em destaque no nome da opção. As janelas, no entanto, quando são mostradas na tela, ficam aquardando a entrada de uma linha de comando. Cada janela possui uma linha de comando específica e, no caso de haver erro na entrada da sua linha de comando, uma mensagem é mostrada na tela.

A tecla ESC serve para abortar qualquer operação do algoritmo, como por exemplo, apagar da tela as mensagens de erro, dando prosseguimento ao processamento normal do programa.

Para executar o algoritmo, basta escrever na linha de comandos do sistema operacional, o nome do programa da camada base (AvalEst). O processamento do algoritmo começa com uma tela de abertura (figura A.1) que apresenta uma janela que faz a leitura do nome do arquivo da Forma Intermediária Estática da especificação analisada (figura A.2).

Figura A.2 - Janela de leitura da F.I. Estática.

Caso ocorra algum problema do tipo, a especificação Estelle não foi compilada ou o arquivo da FI estática gerado pelo compilador Estelle não se encontra no diretório indicado, a ferramenta alerta o usário com uma mensagem de erro, como é mostrado na figura que segue.

Figura A.3 - Indicação de erro na inicialização do arquivo da FI estática.

A descrição dos comandos da ferramenta é mostrada abaixo, segundo a ordem de aparecimento dos "menus" e das janelas na tela, começando com o mais externo (Menu Principal) até esgotar todas as suas opções. Quando aparecer o trecho ESPAÇO na descrição das linhas de comando das janelas, deve-se teclar um único espaço na entrada destas linhas.

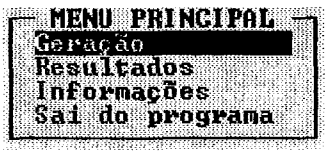

Figura A.4 - O Menu Principal.

a) Menu Principal - Este "menu" apresenta as seguintes opções:

* **G**eração - Faz a abertura do MENU DE GERAÇÃO.

* **R**esultados - Faz a abertura do MENU DE RESULTADOS.

* **I**nformações - Apresenta a janela das informações comentadas anteriormente, na descrição do módulo MostraInformações.

* **S**ai do programa - Termina a execução da ferramenta.

Figura A.5 - Finalização da geração da árvore de alcançabilidade.

b) Menu de Geração - Este "menu" apresenta as seguintes opções:

* Gera **D**ireto - Faz a geração de toda a árvore de alcançabilidade e verifica algumas das propriedades gerais de protocolo (vide figura A.5).

* **P**asso a passo - Faz a abertura do MENU DE PASSO A PASSO.

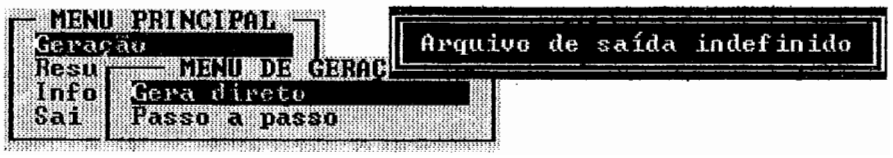

Figura A.6 - Indicação de erro por falta de definição dos arquivos de saída.

Caso se tente realizar a geração antes de definir o nome dos arquivos de saída, tanto da maneira direta quanto na passo a passo, a ferramenta alerta o usuário deste esquecimento (vide figura A.6).

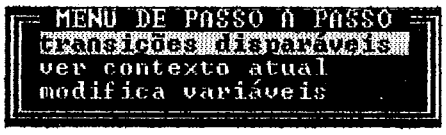

Figura A.7 - O Menu de Passo a Passo.

c) Menu de Passo a passo - Este "menu" apresenta as seguintes opções:

* transições disparáveis - Faz a abertura de um "menu", contendo todas as transições disparáveis no momento. A seleção de uma opção deste "menu", causa o disparo da transição indicada.

* ver contexto atual - Mostra uma série de menus e janelas, contendo todas as informações a respeito da situação atual das instâncias de módulo existentes no momento.

* Modifica Variáveis - Faz a abertura de um "menu", contendo todas as variáveis declaradas no texto da especificação, pelas primitivas $Afetar e $Verificar, do compilador Estelle. Fazendo a seleção de uma dessas variáveis, uma janela é aberta para receber, como entrada, o novo valor da variável escolhida. Este valor deve ser inserido diretamente na janela de acordo com o tipo da variável. No caso da variável ser do tipo BOOLEAN, deve ser fornecido o valor 0 (zero) para o caso FALSE ou o valor 1 (um) para o caso TRUE.

A seguir é mostrada a interface oferecida pela sub-opção "transições disparáveis", que exibe o nome de cada uma das transições aptas a serem disparadas, ou a referência de em qual linha da especificação Estelle a transição se encontra, conforme o exemplo da próxima figura. Os módulos recebem uma numeração de referência e os nomes dos corpos também são apresentados.

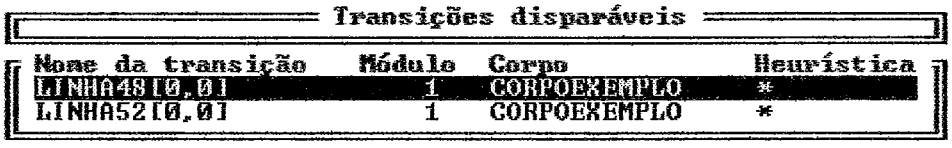

Figura A.8 - Janela de diálogo da sub-opção "transições disparáveis".

d) Menu de Resultados - Este "menu" apresenta as seguintes opções:

Figura A.9 - O Menu de Resultados.

* **C**ontexto atual - O mesmo que a opção "Ver Contexto Atual" do menu de Passo a passo.

* arquivo de **E**ntrada - Abre uma janela para fazer a leitura do nome sem extensão do arquivo que contém uma análise feita anteriormente. A extensão é colocada pela própria ferramenta de acordo com o tipo de arquivo manipulado (.DES, .GRF ou .SEQ).

* arquivo de **S**aída - Abre uma janela para fazer a leitura do nome sem extensão do arquivo em que será gravada a primeira análise realizada. Como na opção anterior, a extensão é inserida pela ferramenta.

* **D**escrição - Mostra o arquivo de descrição dos resultados, cujo nome foi fornecido na opção "Arquivo de Entrada".

* Grafo - Mostra na tela, um estado global contido no arquivo do grafo de execução, cujo nome foi dado na opção "arquivo de entrada". O número do estado global a ser mostrado constitui a linha de comando da janela aberta pela escolha desta opção.

* seQüência - Mostra o arquivo de seqüência de disparos, cujo nome foi fornecido na opção "Arquivo de Entrada".

Figura A.10 - Atribuição de nome aos arquivos de saída.

Ao invés de realizar uma nova geração, o usuário ainda tem a opção de dar o nome dos arquivos de saída do teste que ele já realizou, e que deseja continuar analisando, através da sub-opção "arquivo de entrada". Na fig. A.10 é feita a atribuição de um nome sugestivo para o teste do sistema "Exemplo Inverso" que foi testado na data 12/03/95, sendo que todos os arquivos de saída gerados pelo ferramenta terão o mesmo nome, diferindo apenas na extensão que os classifica.

Figura A.11 - Indicação de que o nome escolhido já existe.

Se o usuário escolher um nome para arquivos de saída, que já exista, o usuário deverá cancelar a operação com ESC ou os arquivos serão gravados por cima com os resultados do novo teste.

A opção "grafo" permite que o usuário carregue qualquer um dos estados globais obtidos pelo algoritmo, como estado atual em análise, para que então, possa estudar seus módulos, verificar a hierarquia dos mesmos, verificar em que estado cada um dos módulos da especificação se encontra, consultar o conteúdo de cada uma das filas, e averiguar o atual valor daquelas variáveis que o compilador tornou visíveis ao usuário, através das diretivas $Afetar e $Verificar. Desta forma, todas as características que compõem o estado global atual, podem ser consultadas em tempo de execução. Estas consultas são exemplificadas nas figuras A.12, 13, 14, 15, 16 e 17.

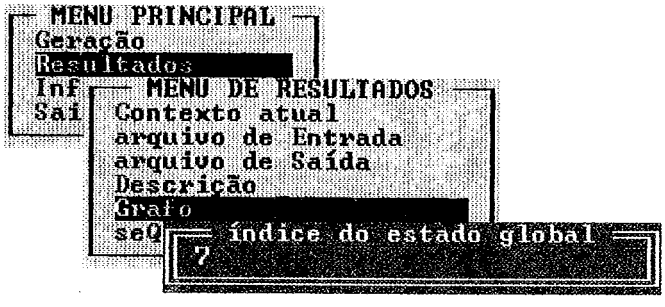

Figura A.12 - Exemplo de carregamento de um novo estado global escolhido.

No sistema do "Exemplo Inverso" foi obtida uma Cadeia de Markov de tempo discreto composta de nove estados globais, sendo que o estado inicial raiz recebe o número um. Caso o usuário tente carregar inadivertidamente um novo estado global com numeração superior a nove, a ferramenta emite uma mensagem de erro, advertindo o usuário da incoerência.

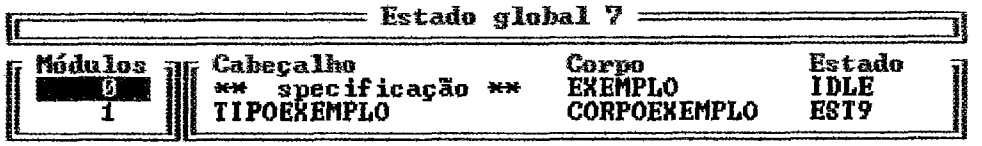

Figura A.13 - Janela de informações a respeito dos módulos do atual estado global.

Figura A.14 - Janela de informações a respeito do contexto do atual estado global.

Figura A.15 - Janela de informações a respeito dos atuais filhos do módulo0.

Figura A.16 - Janela de informações a respeito das filas do módulo 0.

Figura A.17 - Janela de informações a respeito das expressões visíveis do módulo 0.

Na versão para UNIX, a descrição dos estados globais obtidos é apresentada na forma textual, dentro de um quarto arquivo de saída que é gerado com a extensão .EGL, que é uma referência a Estados GLobais.

A qualquer momento da utilização da ferramenta, a opção "Informações" pode ser requisitada, que se abrirá uma janela relatando todas as informações que a ferramenta conseguiu obter do teste e do sistema. Estas informações vão se atualizando ao longo da utilização da ferramenta, entretanto é após a realização de uma geração direta, que este campo fica com o maior número de informações disponíveis (vide figura A.18). Este teste foi realizado num equipamento 486 DX2 66MHz.

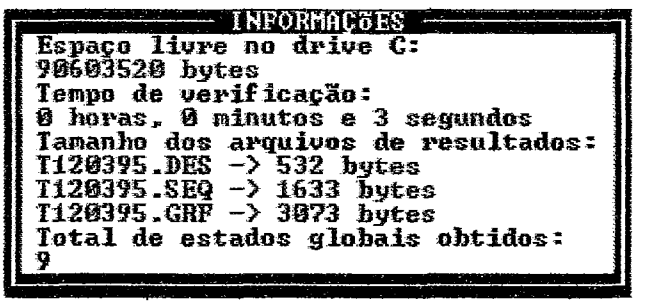

Figura A.18 - Janela de Informações.

A.3 - Gerando um teste

Os comandos descritos na seção anterior, permitem ao usuário operar a ferramenta para fazer a análise de uma única especificação, uma vez que a mudança para uma outra especificação implica na religação de todos os "overlays" da plataforma com o novo módulo da Forma Intermediária dinâmica.

Agora será apresentado um esquema detalhado que auxilia no entendimento do procedimento que se deve ter com alguns dos módulos que compõem a ferramenta desenvolvida. Em seguida será exlicado o que se deve fazer para mudar a especificação em análise na plataforma e gerar um novo teste.

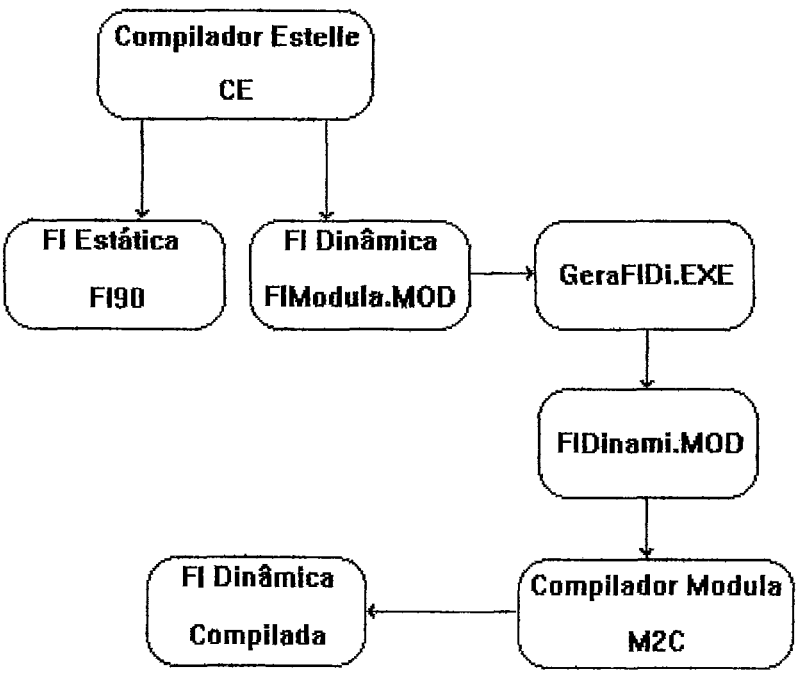

Figura A.19 - Tratamento da Forma Intermediária.

Descrição dos procedimentos para tratar a Forma Intermediária gerada pelo compilador Estelle (vide figura acima).

1) Usar o Compilador Estelle com a nova especificação a fim de gerar os arquivos da FI estática (FI90) e da FI dinâmica (FIMODULA.MOD). A linha de comando do sistema operacional para este passo é dada por:

>CE NomeEsp

,onde NomeEsp é o nome da especificação e > é o "prompt" do sistema.

2) De posse do arquivo FIMODULA.MOD, que constitui a FI dinâmica criada pelo Compilador, executar o programa GERAFIDI.EXE para gerar a Forma Intermediária dinâmica, efetivamente usada pela Plataforma de Software. A saída do programa GERAFIDI.EXE é o arquivo FIDINAMI.MOD obtido a partir de algumas alterações realizadas sobre o arquivo FIMODULA.MOD. A linha de comando do sistema operacional para este passo é dada por:

>GERAFIDI

3) Usar um compilador de MODULA-2 para obter o código objeto do módulo FIDINAMI.MOD.

>M2C FIDINAMI.MOD

4) Religar a camada base e todos os "overlays" da ferramenta para Estelle.

Para que seja possível a compilação do módulo FIDINAMI.MOD, assim como a ligação da ferramenta é necessário ter disponível os arquivos de código objeto de todos os módulos que compõem a plataforma, incluindo o código objeto do módulo FIDINAMI.DEF, encontrado no arquivo FIDINAMI.SYM.

Anexo B

ANEXO B - Especificações e Resultados

Aqui serão expostas as listagens das descrições formais dos sistemas testados pela ferramenta desenvolvida, feitas utilizando a técnica Estelle, assim como os seus respectivos arquivos de saída gerados pelo software que descrevem uma análise sucinta do sistema e ainda apresenta a descrição textual da árvore de alcançabilidade probabilística do mesmo sistema. Esta aplicação exemplifica a utilização da Plataforma de Software.

B.1 O exemplo da "Exclusão Mútua"

a) A Especificação em Estelle do Sistema

```
SPECIFICATION ExclusaoMutua SYSTEMACTIVITY;

DEFAULT INDIVIDUAL QUEUE; TIMESCALE SECONDS;

{**** DECLARACAO DE CANAL ************************************}
CHANNEL CanalDeAcesso (user,provider);
BY user : AcessoReq;
      DadoAtivador;
BY provider : DadoProduzido;
        AcessoNeg;
        AcessoLib;

{**** DECLARACAO DE FUNCAO ************************************}

{**** DECLARACAO DE TIPO DE MODULO ****************************}
MODULE Processo_Type ACTIVITY;
  IP PontoDeAcesso : CanalDeAcesso (user);
END;

{**** DECLARACAO DE TIPO DE CORPO *****************************}
BODY Processo_Body FOR Processo_Type;

  {** Declaracao de Variaveis de Modulo ***********************}
  VAR
    PROBABILITY     : REAL;

    {$Verificar VAR PROBABILITY}

  {** Declaracao de Estados ***********************************}
  STATE
    processando, usandorecurso, esperando;
```

```
  {** Inicializacao de Estado ***************************************}
  INITIALIZE TO processando
  BEGIN
    PROBABILITY := 1;
  END;

  {** Transicoes ****************************************************}
  TRANS
   FROM processando
     TO esperando
     NAME proce1:
     BEGIN
       PROBABILITY := 0.5;
       OUTPUT PontoDeAcesso.AcessoReq;
     END;
   {*************************************************************}
   FROM esperando
     WHEN PontoDeAcesso.AcessoNeg
     TO esperando
     NAME proce2:
     BEGIN
       PROBABILITY := 0.5;
       OUTPUT PontoDeAcesso.AcessoReq;
     END;
     {*************************************************************}
     WHEN PontoDeAcesso.AcessoLib
     TO usandorecurso
     NAME proce3:
     BEGIN
       PROBABILITY := 0.5;
       OUTPUT PontoDeAcesso.DadoAtivador;
     END;
   {*************************************************************}
   FROM usandorecurso
     WHEN PontoDeAcesso.DadoProduzido
     TO processando
     NAME proce4:
     BEGIN
       PROBABILITY := 0.5;
     END;

  END; (* de Processo_Body *)

{**** DECLARACAO DE TIPO DE MODULO ***************************}
MODULE Recurso_Type activity;
  IP PontoDeAcesso : ARRAY[0..1] OF CanalDeAcesso (provider);
END;

{**** DECLARACAO DE TIPO DE CORPO ****************************}
BODY Recurso_Body FOR Recurso_Type;

  {** Declaracao de Variaveis de Modulo *************************}
  VAR
    PROBABILITY : REAL;

    {$Verificar VAR PROBABILITY}

  {** Declaracao de Estados**************************************}
  STATE
    livre, ocupado;
```

```
{** Inicializacao de Estado **********************************************}
INITIALIZE TO livre
BEGIN
  PROBABILITY := 1;
END;

{** Transicoes ***********************************************************}
TRANS
 FROM livre
   WHEN PontoDeAcesso[0].AcessoReq
   TO ocupado
   NAME recurso1:
   BEGIN
     PROBABILITY := 0.5;
     OUTPUT PontoDeAcesso[0].AcessoLib;
   END;
   {*******************************************************}
   WHEN PontoDeAcesso[1].AcessoReq
   TO ocupado
   NAME recurso2:
   BEGIN
     PROBABILITY := 0.5;
      OUTPUT PontoDeAcesso[1].AcessoLib;
   END;
 {*******************************************************}
 FROM ocupado
   WHEN PontoDeAcesso[0].AcessoReq
   TO ocupado
   NAME recurso3:
   BEGIN
     PROBABILITY := 0.5;
     OUTPUT PontoDeAcesso[0].AcessoNeg;
   END;
   {*******************************************************}
   WHEN PontoDeAcesso[1].AcessoReq
   TO ocupado
   NAME recurso4:
   BEGIN
     PROBABILITY := 0.5;
     OUTPUT PontoDeAcesso[1].AcessoNeg;
   END;
   {*******************************************************}
   WHEN PontoDeAcesso[0].DadoAtivador
   TO livre
   NAME recurso5:
   BEGIN
     PROBABILITY := 0.5;
     OUTPUT PontoDeAcesso[0].DadoProduzido;
   END;
   {*******************************************************}
   WHEN PontoDeAcesso[1].DadoAtivador
   TO livre
   NAME recurso6:
   BEGIN
     PROBABILITY := 0.5;
     OUTPUT PontoDeAcesso[1].DadoProduzido;
   END;
   {*******************************************************}
```

```
   END; (* de Recurso_Body *)

{**** DECLARACAO DE MODULOS ***********************************}
MODVAR
  Proc0,Proc1 : Processo_Type;
  Rec         : Recurso_Type;

{**** INICIALIZACAO DE MODULOS ********************************}
INITIALIZE BEGIN
  {** Atribuicao de Corpos ***********************************}
  INIT Rec WITH Recurso_Body;
  INIT Proc0 WITH Processo_Body;
  INIT Proc1 WITH Processo_Body;
  {** Conexao de Pontos de Interacao *************************}
  CONNECT Proc0.PontoDeAcesso TO Rec.PontoDeAcesso[0];
  CONNECT Proc1.PontoDeAcesso TO Rec.PontoDeAcesso[1];
END;

{************************************************************}
END. (* Fim da Especificacao ExclusaoMutua *)
```

b) A Descrição das Propriedades do Sistema

```
Especificação: EXCLUSAOMUTUA
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Verificação: PROPRIEDADES
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Expressões não usadas: 2
                1
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Razão do final: Geração completada
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Conclusão:

Total de estados globais obtidos: 31
Tempo de verificação: 0 horas, 1 minutos e 32 segundos
Especificação limitada e reinicializável
```

c) As Seqüências de Transições da Árvore de Alcançabilidade

```
Estado raiz = (1)
(1) -Modulo 2 -Transição PROCE1[0,0] -Prob 0.50000 --> (2)
(2) -Modulo 1 -Transição RECURSO1 -Prob 0.50000 --> (3)
(3) -Modulo 2 -Transição PROCE3 -Prob 0.50000 --> (4)
(4) -Modulo 3 -Transição PROCE1[0,0] -Prob 0.50000 --> (5)
(5) -Modulo 1 -Transição RECURSO5 -Prob 0.50000 --> (6)
(6) -Modulo 1 -Transição RECURSO2 -Prob 0.50000 --> (7)
(7) -Modulo 2 -Transição PROCE4 -Prob 0.50000 --> (8)
(8) -Modulo 2 -Transição PROCE1[0,0] -Prob 0.50000 --> (9)
(9) -Modulo 1 -Transição RECURSO3 -Prob 0.50000 --> (10)
(10) -Modulo 3 -Transição PROCE3 -Prob 0.50000 --> (11)
(11) -Modulo 1 -Transição RECURSO6 -Prob 0.50000 --> (12)
(12) -Modulo 3 -Transição PROCE4 -Prob 0.50000 --> (13)
```

(13) -Modulo 3 -Transição PROCE1[0,0] -Prob 0.50000 --> (14)
(14) -Modulo 1 -Transição RECURSO2 -Prob 0.50000 --> (10)
** BACKTRACK TO (14) **
(14) -Modulo 2 -Transição PROCE2 -Prob 0.50000 --> (15)
(15) -Modulo 1 -Transição RECURSO1 -Prob 0.50000 --> (16)
(16) -Modulo 1 -Transição RECURSO4 -Prob 0.50000 --> (17)
(17) -Modulo 2 -Transição PROCE3 -Prob 0.50000 --> (18)
(18) -Modulo 3 -Transição PROCE2 -Prob 0.50000 --> (5)
** BACKTRACK TO (18) **
(18) -Modulo 1 -Transição RECURSO5 -Prob 0.50000 --> (19)
(19) -Modulo 2 -Transição PROCE4 -Prob 0.50000 --> (20)
(20) -Modulo 3 -Transição PROCE2 -Prob 0.50000 --> (21)
(21) -Modulo 2 -Transição PROCE1[0,0] -Prob 0.50000 --> (15)
** BACKTRACK TO (21) **
(21) -Modulo 1 -Transição RECURSO2 -Prob 0.50000 --> (8)
** BACKTRACK TO (21) **
(21) --VOLTA PARA--> (20)
(20) -Modulo 2 -Transição PROCE1[0,0] -Prob 0.50000 --> (22)
(22) -Modulo 1 -Transição RECURSO1 -Prob 0.50000 --> (17)
** BACKTRACK TO (22) **
(22) -Modulo 3 -Transição PROCE2 -Prob 0.50000 --> (15)
** BACKTRACK TO (22) **
(22) --VOLTA PARA--> (20)
(20) --VOLTA PARA--> (19)
(19) -Modulo 3 -Transição PROCE2 -Prob 0.50000 --> (6)
** BACKTRACK TO (19) **
(19) --VOLTA PARA--> (18)
(18) --VOLTA PARA--> (17)
(17) -Modulo 3 -Transição PROCE2 -Prob 0.50000 --> (16)
** BACKTRACK TO (17) **
(17) --VOLTA PARA--> (16)
(16) -Modulo 2 -Transição PROCE3 -Prob 0.50000 --> (5)
** BACKTRACK TO (16) **
(16) --VOLTA PARA--> (15)
(15) -Modulo 1 -Transição RECURSO2 -Prob 0.50000 --> (9)
** BACKTRACK TO (15) **
(15) --VOLTA PARA--> (14)
(14) --VOLTA PARA--> (13)
(13) -Modulo 2 -Transição PROCE2 -Prob 0.50000 --> (2)
** BACKTRACK TO (13) **
(13) --VOLTA PARA--> (12)
(12) -Modulo 2 -Transição PROCE2 -Prob 0.50000 --> (23)
(23) -Modulo 1 -Transição RECURSO1 -Prob 0.50000 --> (24)
(24) -Modulo 2 -Transição PROCE3 -Prob 0.50000 --> (25)
(25) -Modulo 3 -Transição PROCE4 -Prob 0.50000 --> (4)
** BACKTRACK TO (25) **
(25) -Modulo 1 -Transição RECURSO5 -Prob 0.50000 --> (26)
(26) -Modulo 2 -Transição PROCE4 -Prob 0.50000 --> (27)
(27) -Modulo 2 -Transição PROCE1[0,0] -Prob 0.50000 --> (23)
** BACKTRACK TO (27) **
(27) -Modulo 3 -Transição PROCE4 -Prob 0.50000 --> (1)
** BACKTRACK TO (27) **
(27) --VOLTA PARA--> (26)
(26) -Modulo 3 -Transição PROCE4 -Prob 0.50000 --> (28)
(28) -Modulo 2 -Transição PROCE4 -Prob 0.50000 --> (1)
** BACKTRACK TO (28) **
(28) -Modulo 3 -Transição PROCE1[0,0] -Prob 0.50000 --> (6)
** BACKTRACK TO (28) **
(28) --VOLTA PARA--> (26)
(26) --VOLTA PARA--> (25)

(25) --VOLTA PARA--> (24)
(24) -Modulo 3 -Transição PROCE4 -Prob 0.50000 --> (3)
** BACKTRACK TO (24) **
(24) --VOLTA PARA--> (23)
(23) -Modulo 3 -Transição PROCE4 -Prob 0.50000 --> (2)
** BACKTRACK TO (23) **
(23) --VOLTA PARA--> (12)
(12) --VOLTA PARA--> (11)
(11) -Modulo 2 -Transição PROCE2 -Prob 0.50000 --> (29)
(29) -Modulo 1 -Transição RECURSO3 -Prob 0.50000 --> (11)
** BACKTRACK TO (29) **
(29) -Modulo 1 -Transição RECURSO6 -Prob 0.50000 --> (23)
** BACKTRACK TO (29) **
(29) --VOLTA PARA--> (11)
(11) --VOLTA PARA--> (10)
(10) -Modulo 2 -Transição PROCE2 -Prob 0.50000 --> (9)
** BACKTRACK TO (10) **
(10) --VOLTA PARA--> (9)
(9) -Modulo 3 -Transição PROCE3 -Prob 0.50000 --> (29)
** BACKTRACK TO (9) **
(9) --VOLTA PARA--> (8)
(8) -Modulo 3 -Transição PROCE3 -Prob 0.50000 --> (30)
(30) -Modulo 2 -Transição PROCE1[0,0] -Prob 0.50000 --> (29)
** BACKTRACK TO (30) **
(30) -Modulo 1 -Transição RECURSO6 -Prob 0.50000 --> (27)
** BACKTRACK TO (30) **
(30) --VOLTA PARA--> (8)
(8) --VOLTA PARA--> (7)
(7) -Modulo 3 -Transição PROCE3 -Prob 0.50000 --> (31)
(31) -Modulo 1 -Transição RECURSO6 -Prob 0.50000 --> (26)
** BACKTRACK TO (31) **
(31) -Modulo 2 -Transição PROCE4 -Prob 0.50000 --> (30)
** BACKTRACK TO (31) **
(31) --VOLTA PARA--> (7)
(7) --VOLTA PARA--> (6)
(6) -Modulo 2 -Transição PROCE4 -Prob 0.50000 --> (21)
** BACKTRACK TO (6) **
(6) --VOLTA PARA--> (5)
(5) -Modulo 1 -Transição RECURSO4 -Prob 0.50000 --> (18)
** BACKTRACK TO (5) **
(5) --VOLTA PARA--> (4)
(4) -Modulo 1 -Transição RECURSO5 -Prob 0.50000 --> (28)
** BACKTRACK TO (4) **
(4) --VOLTA PARA--> (3)
(3) -Modulo 3 -Transição PROCE1[0,0] -Prob 0.50000 --> (16)
** BACKTRACK TO (3) **
(3) --VOLTA PARA--> (2)
(2) -Modulo 3 -Transição PROCE1[0,0] -Prob 0.50000 --> (15)
** BACKTRACK TO (2) **
(2) --VOLTA PARA--> (1)
(1) -Modulo 3 -Transição PROCE1[0,0] -Prob 0.50000 --> (21)
** BACKTRACK TO (1) **
** árvore de alcançabilidade esgotada **

B.2 O exemplo "Exemplo"

a) A Especificação em Estelle do Sistema

```
SPECIFICATION Exemplo SYSTEMACTIVITY;

DEFAULT INDIVIDUAL QUEUE;
TIMESCALE SECONDS;

{**** DECLARACAO DE CONSTANTES ********************************}

CONST low =0.05; (* probabilidade menor *)
    high=0.95; (* probabilidade maior *)


{**** DECLARACAO DE TIPO DE MODULO ****************************}

  MODULE TipoExemplo ACTIVITY;
  END;

{**** DECLARACAO DE CORPO *************************************}

  BODY CorpoExemplo FOR TipoExemplo;

    {*** Declaracao de Estados ********************************}
    STATE   Est1,
            Est2,
            Est3,
            Est4,
            Est5,
            Est6,
            Est7,
            Est8,
            Est9;
    (* Como o sistema possui apenas um modulo c/ uma instancia *)
    (* os estados globais serao os proprios estados locais    *)

    {*** Declaracao de variavel *******************************}
    VAR PROBABILITY: REAL; (* probabilidade de disparo de transicao *)
    {$verificar VAR PROBABILITY}

    {*** Inicializacao ****************************************}
    INITIALIZE TO Est1
    BEGIN
    PROBABILITY:= 1;
    END;

    {*** Transicoes *******************************************}
    TRANS
      FROM Est1
        TO Est2
          BEGIN
            PROBABILITY:= high;
          END;
        TO Est3
```

```
    BEGIN
      PROBABILITY:= low;
    END;
{*********************************************************}
FROM Est2
  TO Est4
    BEGIN
      PROBABILITY:= high;
    END;
  TO Est5
    BEGIN
      PROBABILITY:= low;
    END;
{*********************************************************}
FROM Est3
  TO Est5
    BEGIN
      PROBABILITY:= high;
    END;
  TO Est6
    BEGIN
      PROBABILITY:= low;
    END;
{*********************************************************}
FROM Est4
  TO Est2
    BEGIN
      PROBABILITY:= high;
    END;
  TO Est7
    BEGIN
      PROBABILITY:= low;
    END;
{*********************************************************}
FROM Est5
  TO Est7
    BEGIN
      PROBABILITY:= high;
    END;
  TO Est8
    BEGIN
      PROBABILITY:= low;
    END;
{*********************************************************}
FROM Est6
  TO Est8
    BEGIN
      PROBABILITY:= high;
    END;
  TO Est3
    BEGIN
      PROBABILITY:= low;
    END;
{*********************************************************}
FROM Est7
  TO Est4
    BEGIN
      PROBABILITY:= high;
    END;
  TO Est9
```

```
        BEGIN
          PROBABILITY:= low;
        END;
      {*******************************************************}
      FROM Est8
        TO Est9
          BEGIN
            PROBABILITY:= high;
          END;
        TO Est6
          BEGIN
            PROBABILITY:= low;
          END;
      {*******************************************************}
      FROM Est9
        TO Est1
          BEGIN
            PROBABILITY:= high;
          END;
        TO Est5
          BEGIN
            PROBABILITY:= low;
          END;
      {*******************************************************}

  END; (* fim do corpo *)

{**** DECLARACAO DE MODULO ********************************}

  MODVAR InstanciaUnica : TipoExemplo;

{**** INICIALIZACAO ***************************************}

  INITIALIZE BEGIN
    (* inicializa modulo com respectivo corpo *)
    INIT InstanciaUnica WITH CorpoExemplo;
  END;

{*******************************************************}

END. (* Fim da Especificacao *)
```

b) A Descrição das Propriedades do Sistema

```
Especificação: EXEMPLO
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Verificação: PROPRIEDADES
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Expressões não usadas: 1
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Razão do final: Geração completada
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Conclusão:

Total de estados globais obtidos: 9
Tempo de verificação: 0 horas, 0 minutos e 15 segundos
Especificação limitada e reinicializável
```

c) As Seqüências de Transições da Árvore de Alcançabilidade

```
Estado raiz = (1)
 (1) -Modulo 1 -Transição LINHA52[0,0] -Prob 0.05000 --> (2)
 (2) -Modulo 1 -Transição LINHA68[0,0] -Prob 0.95000 --> (3)
 (3) -Modulo 1 -Transição LINHA88[0,0] -Prob 0.95000 --> (4)
 (4) -Modulo 1 -Transição LINHA108[0,0] -Prob 0.95000 --> (5)
 (5) -Modulo 1 -Transição LINHA78[0,0] -Prob 0.95000 --> (6)
 (6) -Modulo 1 -Transição LINHA62[0,0] -Prob 0.05000 --> (3)
** BACKTRACK TO (6) **
 (6) -Modulo 1 -Transição LINHA58[0,0] -Prob 0.95000 --> (5)
** BACKTRACK TO (6) **
 (6) --VOLTA PARA--> (5)
 (5) -Modulo 1 -Transição LINHA82[0,0] -Prob 0.05000 --> (4)
** BACKTRACK TO (5) **
 (5) --VOLTA PARA--> (4)
 (4) -Modulo 1 -Transição LINHA112[0,0] -Prob 0.05000 --> (7)
 (7) -Modulo 1 -Transição LINHA132[0,0] -Prob 0.05000 --> (3)
** BACKTRACK TO (7) **
 (7) -Modulo 1 -Transição LINHA128[0,0] -Prob 0.95000 --> (1)
** BACKTRACK TO (7) **
 (7) --VOLTA PARA--> (4)
 (4) --VOLTA PARA--> (3)
 (3) -Modulo 1 -Transição LINHA92[0,0] -Prob 0.05000 --> (8)
 (8) -Modulo 1 -Transição LINHA118[0,0] -Prob 0.95000 --> (7)
** BACKTRACK TO (8) **
 (8) -Modulo 1 -Transição LINHA122[0,0] -Prob 0.05000 --> (9)
 (9) -Modulo 1 -Transição LINHA102[0,0] -Prob 0.05000 --> (2)
** BACKTRACK TO (9) **
 (9) -Modulo 1 -Transição LINHA98[0,0] -Prob 0.95000 --> (8)
** BACKTRACK TO (9) **
 (9) --VOLTA PARA--> (8)
 (8) --VOLTA PARA--> (3)
 (3) --VOLTA PARA--> (2)
 (2) -Modulo 1 -Transição LINHA72[0,0] -Prob 0.05000 --> (9)
** BACKTRACK TO (2) **
 (2) --VOLTA PARA--> (1)
 (1) -Modulo 1 -Transição LINHA48[0,0] -Prob 0.95000 --> (6)
** BACKTRACK TO (1) **
** árvore de alcançabilidade esgotada **
```

B.3 O exemplo do "Produtor e o Consumidor"

a) A Especificação em Estelle do Sistema

```
SPECIFICATION ProdutorConsumidor SYSTEMACTIVITY;

DEFAULT INDIVIDUAL QUEUE; TIMESCALE SECONDS;

{**** DECLARACAO DE CANAL ************************************}
CHANNEL Interface_Type (user,provider);
BY user : Msg;
```

```
BY provider : Lib;

{**** DECLARACAO DE TIPO DE MODULO ************************}
MODULE Produtor_Type ACTIVITY;
  IP Interface : Interface_Type (user);
END;

{**** DECLARACAO DE TIPO DE CORPO *************************}
BODY Produtor_Body FOR Produtor_Type;

  {** Declaracao de Variaveis de Modulo ********************}
  VAR
    PROBABILITY      : REAL;

    {$Verificar VAR PROBABILITY}

  {** OBS : Nao foram definidos os Estados Locais do Modulo ***************}

  {** Inicializacao de Modulo ******************************************}
  INITIALIZE BEGIN
    PROBABILITY := 1;
  END;

  {** Transicoes ******************************************************}
  TRANS
    WHEN Interface.Lib
    NAME Produtor1:
    BEGIN
      PROBABILITY := 1;
      OUTPUT Interface.Msg;
    END;

END; (* Produtor_Body *)

{**** DECLARACAO DE TIPO DE MODULO ************************}
MODULE Consumidor_Type activity;
  IP Interface : Interface_Type (provider);
END;

{**** DECLARACAO DE TIPO DE CORPO *************************}
BODY Consumidor_Body FOR Consumidor_Type;

  {** Declaracao de Variaveis de Modulo ********************}
  VAR
    Flag : BOOLEAN;
    PROBABILITY : REAL;

    {$Verificar VAR PROBABILITY}

  {** OBS : Nao foram definidos os Estados Locais do Modulo ***************}

  {** Inicializacao de Estado ******************************************}
  INITIALIZE BEGIN
    Flag := TRUE;
    PROBABILITY := 1;
  END;

  {** Transicoes ******************************************************}
  TRANS
    PROVIDED Flag = TRUE
```

```
    NAME Consumidor1:
    BEGIN
      PROBABILITY := 1;
      OUTPUT Interface.Lib;
      Flag := FALSE;
    END;
  {**************************************************}
  TRANS
    WHEN Interface.Msg
    NAME Consumidor2:
    BEGIN
      PROBABILITY := 1;
      OUTPUT Interface.Lib;
    END;

END; (* Consumidor_Body *)

{**** DECLARACAO DE MODULOS ********************************}
MODVAR
  Produtor   : Produtor_Type;
  Consumidor : Consumidor_Type;

{**** INICIALIZACAO DE MODULOS *****************************}
INITIALIZE BEGIN

  {** Atribuicao de Corpos ********************************}
  INIT Consumidor WITH Consumidor_Body;
  INIT Produtor   WITH Produtor_Body;

  {** Conexao de Pontos de Interacao ***********************}
  CONNECT Produtor.Interface TO Consumidor.Interface;

END;

{**************************************************}
END. (* Fim da Especificacao ProdutorConsumidor *)
```

b) A Descrição das Propriedades do Sistema

```
Especificação: PRODUTORCONSUMID
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Verificação: PROPRIEDADES
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Expressões não usadas: 2
                1
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Razão do final: Geração completada
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Conclusão:

Total de estados globais obtidos: 3
Tempo de verificação: 0 horas, 0 minutos e 3 segundos
Especificação limitada e não reinicializável
```

c) As Seqüências de Transições da Árvore de Alcançabilidade

```
Estado raiz = (1)
 (1) -Modulo 1 -Transição CONSUMIDOR1[0,0] -Prob 1.00000 --> (2)
 (2) -Modulo 2 -Transição PRODUTOR1 -Prob 1.00000 --> (3)
 (3) -Modulo 1 -Transição CONSUMIDOR2 -Prob 1.00000 --> (2)
** BACKTRACK TO (3) **
 (3) --VOLTA PARA--> (2)
 (2) --VOLTA PARA--> (1)
** árvore de alcançabilidade esgotada **
```

B.4 O exemplo "Sem Probabilidade"

a) A Especificação em Estelle do Sistema

```
SPECIFICATION ProdutorConsumidor SYSTEMACTIVITY;

DEFAULT INDIVIDUAL QUEUE; TIMESCALE SECONDS;

{**** DECLARACAO DE CANAL ************************************}
CHANNEL Interface_Type (user,provider);
BY user : Msg;
BY provider : Lib;

{**** DECLARACAO DE TIPO DE MODULO ***************************}
MODULE Produtor_Type ACTIVITY;
  IP Interface : Interface_Type (user);
END;

{**** DECLARACAO DE TIPO DE CORPO ****************************}
BODY Produtor_Body FOR Produtor_Type;

  {** Declaracao de Variaveis de Modulo **********************}
  VAR
    PROBABILITY     : REAL;

    {$Verificar VAR PROBABILITY}

  {** OBS : Nao foram definidos os Estados Locais do Modulo **********}

  {** Inicializacao de Modulo ********************************}
  INITIALIZE BEGIN
    PROBABILITY := 1;
  END;

  {** Transicoes *********************************************}
  TRANS
    WHEN Interface.Lib
    NAME Produtor1:
    BEGIN
      (*PROBABILITY := 1;*)
      OUTPUT Interface.Msg;
```

```
    END;

END; (* Produtor_Body *)

{**** DECLARACAO DE TIPO DE MODULO ***************************}
MODULE Consumidor_Type activity;
  IP Interface : Interface_Type (provider);
END;

{**** DECLARACAO DE TIPO DE CORPO ****************************}
BODY Consumidor_Body FOR Consumidor_Type;

  {** Declaracao de Variaveis de Modulo ***********************}
  VAR
    Flag : BOOLEAN;
    PROBABILITY : REAL;

    {$Verificar VAR PROBABILITY}

  {** OBS : Nao foram definidos os Estados Locais do Modulo ***************}

  {** Inicializacao de Estado *********************************}
  INITIALIZE BEGIN
    Flag := TRUE;
    PROBABILITY := 1;
  END;

  {** Transicoes **********************************************}
  TRANS
    PROVIDED Flag = TRUE
    NAME Consumidor1:
    BEGIN
      (*PROBABILITY := 1;*)
      OUTPUT Interface.Lib;
      Flag := FALSE;
    END;
  {****************************************************}
  TRANS
    WHEN Interface.Msg
    NAME Consumidor2:
    BEGIN
      (*PROBABILITY := 1;*)
      OUTPUT Interface.Lib;
    END;

END; (* Consumidor_Body *)

{**** DECLARACAO DE MODULOS ***********************************}
MODVAR
  Produtor   : Produtor_Type;
  Consumidor : Consumidor_Type;

{**** INICIALIZACAO DE MODULOS ********************************}
INITIALIZE BEGIN

  {** Atribuicao de Corpos ************************************}
  INIT Consumidor WITH Consumidor_Body;
  INIT Produtor   WITH Produtor_Body;

  {** Conexao de Pontos de Interacao **************************}
```

```
  CONNECT Produtor.Interface TO Consumidor.Interface;

END;

{*************************************************************}
END. (* Fim da Especificacao ProdutorConsumidor *)
```

b) A Descrição das Propriedades do Sistema

```
Especificação: PRODUTORCONSUMID
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Verificação: PROPRIEDADES
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Expressões não usadas: 2
                  1
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Razão do final: Geração completada
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Conclusão:

Total de estados globais obtidos: 3
Tempo de verificação: 0 horas, 0 minutos e 3 segundos
Especificação limitada e não reinicializável
```

c) As Seqüências de Transições da Árvore de Alcançabilidade

```
Estado raiz = (1)
 (1) -Modulo 1 -Transição CONSUMIDOR1[0,0] -Prob 1.00000 --> (2)
 (2) -Modulo 2 -Transição PRODUTOR1 -Prob 1.00000 --> (3)
 (3) -Modulo 1 -Transição CONSUMIDOR2 -Prob 1.00000 --> (2)
** BACKTRACK TO (3) **
 (3) --VOLTA PARA--> (2)
 (2) --VOLTA PARA--> (1)
** árvore de alcançabilidade esgotada **
```

B.5 O exemplo "Stop and Wait"

a) A Especificação em Estelle do Sistema

```
SPECIFICATION StopWait SYSTEMACTIVITY;

TIMESCALE SECONDS;

{**** DECLARACAO DE CONSTANTE ************************************}
CONST Zero = 0;
    Um  = 1;
```

```
{**** DECLARACAO DE TIPO ****************************************}
TYPE Seq_type  = Zero,.Um;
   Id_type   = (Data,Ack);
   Ndata_type = RECORD
              Id : Id_type;
              Seq : Seq_type;
            END;

{**** DECLARACAO DE CANAL ***************************************}
CHANNEL Clock_Access (user,provider);
BY user : TimeOut;
      CanConf;
BY provider : CancelTimer;
        StartTimer;

CHANNEL N_Access (user,provider);
BY user : DataRequest (NData : NData_type);
BY provider : DataResponse (NData : NData_type);

{**** DECLARACAO DE TIPO DE MODULO ******************************}
MODULE Timer_Type ACTIVITY;
  IP c : Clock_Access (user) COMMON QUEUE;
END;

{************************************************************}
MODULE Transmiter_Type ACTIVITY;
IP c : Clock_Access (provider) COMMON QUEUE;
  n : N_Access (user) INDIVIDUAL QUEUE;
END;

{************************************************************}
MODULE Receiver_Type ACTIVITY;
  IP N : N_Access (user) INDIVIDUAL QUEUE;
END;

{************************************************************}
MODULE Network_Type ACTIVITY;
  IP n : ARRAY [Zero..Um] OF N_Access (provider) INDIVIDUAL QUEUE;
END;

{**** DECLARACAO DE TIPO DE CORPO *******************************}
BODY Timer_Body FOR Timer_Type;

  {** Declaracao de Constante ***********************************}
  CONST Del = 5000; (* este DELAY representa o num. meio de ticks do pacote*)

  {** Declaracao de Variavel ************************************}
  VAR
    PROBABILITY : REAL;

    {$Verificar VAR PROBABILITY}

  {** Declaracao de Estados *************************************}
  STATE
    Stop, Start;

  {** Inicializacao de Estado ***********************************}
  INITIALIZE TO Stop
  BEGIN
    PROBABILITY := 1;
```

```
  END;

  {** Transicoes ***********************************************}
  TRANS
   FROM Stop TO Stop
     WHEN c.CancelTimer
     NAME Time1:
     BEGIN
       OUTPUT c.CanConf;
     END;
   {*********************************************************}
   FROM Stop TO Start
     WHEN c.StartTimer
     NAME Time2:
     BEGIN END;
   {*********************************************************}
   FROM Start TO Stop
     WHEN c.CancelTimer
     NAME Time3:
     BEGIN
       OUTPUT c.CanConf;
     END;
   {*********************************************************}
   FROM Start TO Start
     WHEN c.StartTimer
     NAME Time4:
     BEGIN END;
   {*********************************************************}
   FROM Start TO Stop
     DELAY (Del)
     NAME Time5:
     BEGIN
       OUTPUT c.TimeOut;
     END;

  END; (* de Timer_Body *)

{***********************************************************}
BODY Network_Body FOR Network_Type;

  {** Declaracao de Constante ***********************************}
  CONST pl = 0.001;

  {** Declaracao de Variaveis de Modulo *************************}
  VAR
    PROBABILITY : REAL;
    b : NData_type;

    {$Verificar VAR PROBABILITY}

  {** Inicializacao de Estado ***********************************}
  INITIALIZE
  BEGIN
    PROBABILITY := 1;
  END;

  {** Transicoes ************************************************}
  TRANS
    WHEN n[0].DataRequest
    PROVIDED (NData.Id = Data)
```

```
    NAME Network1:
    BEGIN
      PROBABILITY := pl;
      b.Seq := 1;
      b.Id := NData.Id;
      OUTPUT n[1].DataResponse(b);
    END;
    {*********************************************************}
    PROVIDED (NData.Id = Data)
    NAME Network2:
    BEGIN
      PROBABILITY := pl;
      OUTPUT n[1].DataResponse(NData);
    END;
    {*********************************************************}
    WHEN n[1].DataRequest
    PROVIDED (NData.Id = Ack)
    NAME Network3:
    BEGIN
      OUTPUT n[0].DataResponse(NData);
    END;

  END; (* de Network_Body *)

{*************************************************************}
BODY Transmiter_Body FOR Transmiter_Type;

  {** Declaracao de Constante ***************************************}
  CONST Del = 500;

  {** Declaracao de Variaveis de Modulo *****************************}
  VAR
    PROBABILITY : REAL;
    b : NData_type;
    Send_seq : Seq_type;

    {$Verificar VAR PROBABILITY}

  {** Declaracao de Estados *****************************************}
  STATE
    Wfcc,
    Transmitindo,
    EsperandoAck,
    Estab,
    Erro;
  STATESET
    Either = [Wfcc, EsperandoAck, Estab, Erro];

  {** Inicializacao de Estado ***************************************}
  INITIALIZE TO Estab
  BEGIN
    PROBABILITY := 1;
    Send_seq := 0;
  END;

  {** Transicoes ****************************************************}
  TRANS
   FROM Estab TO Transmitindo
     NAME Transmit1:
     BEGIN
```

```
      OUTPUT c.StartTimer;
    END;
  {*************************************************************}
  FROM Transmitindo TO EsperandoAck
    DELAY (Del)
    NAME Transmit2:
    BEGIN
      b.Id := Data;
      b.Seq:= 0;
      OUTPUT n.DataRequest(b);
    END;
  {*************************************************************}
  FROM Either TO Erro
    WHEN c.TimeOut
    NAME Transmit3:
    BEGIN END;
  {*************************************************************}
  FROM EsperandoAck TO Wfcc
    WHEN n.DataResponse
    PROVIDED (NData.Id = Ack) AND (NData.Seq = 0)
    NAME Transmit4:
    BEGIN
      OUTPUT c.CancelTimer;
    END;
  {*************************************************************}
  FROM EsperandoAck TO EsperandoAck
    WHEN n.DataResponse
    PROVIDED (NData.Id = Ack) AND (NData.Seq = 1)
    NAME Transmit5:
    BEGIN
      OUTPUT n.DataRequest(b);
    END;
  {*************************************************************}
  FROM Wfcc TO Estab
    WHEN c.CanConf
    NAME Transmit6:
    BEGIN END;

END; (* de Transmiter_Body *)

{*****************************************************************}
BODY Receiver_Body FOR Receiver_Type;

  {** Declaracao de Constante ***************************************}
  CONST Pros = 500;

  {** Declaracao de Variaveis de Modulo *****************************}
  VAR
    PROBABILITY : REAL;
    b : NData_type;
    Rec_seq : Seq_type;

    {$Verificar VAR PROBABILITY}

  {** Declaracao de Estados *****************************************}
  STATE
    Idle,
    Processando;

  {** Inicializacao de Estado ***************************************}
```

```
  INITIALIZE TO Idle
  BEGIN
    PROBABILITY := 1;
    Rec_seq := 0;
  END;

  {** Transicoes ************************************************}
  TRANS
   FROM Idle TO Processando
     WHEN n.DataResponse
     PROVIDED NData.Id = Data
     NAME Recebe1:
     BEGIN
       Rec_seq := NData.Seq;
     END;
   {*********************************************************}
   FROM Processando TO Idle
     PROVIDED Rec_seq = 0
     DELAY (Pros)
     NAME Recebe2:
     BEGIN
       b.Id := Ack;
       b.Seq:= Rec_seq;
       OUTPUT n.DataRequest(b);
     END;
   {*********************************************************}
   FROM Processando TO Idle
     PROVIDED Rec_seq = 0
     DELAY (Pros)
     NAME Recebe3:
     BEGIN
       b.Id := Ack;
       b.Seq:= Rec_seq;
       OUTPUT n.DataRequest(b);
     END;

  END; (* de Receiver_Body *)

{**** DECLARACAO DE MODULOS ***********************************}
MODVAR
  Timer     : Timer_Type;
  Transmiter : Transmiter_Type;
  Network    : Network_Type;
  Receiver   : Receiver_Type;

{**** INICIALIZACAO DE MODULOS ********************************}
INITIALIZE BEGIN
  {** Atribuicao de Corpos ***************************************}
  INIT Network WITH Network_Body;
  INIT Receiver WITH Receiver_Body;
  INIT Timer WITH Timer_Body;
  INIT Transmiter WITH Transmiter_Body;
  {** Conexao de Pontos de Interacao *****************************}
  CONNECT Timer.c TO Transmiter.c;
  CONNECT Transmiter.n TO Network.n[0];
  CONNECT Receiver.n TO Network.n[1];
END;

{*********************************************************}
END. (* Fim da Especificacao StopWait *)
```

b) A Descrição das Propriedades do Sistema

```
Especificação: STOPWAIT
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Verificação: PROPRIEDADES
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Expressões não usadas: 4
                       3
                       2
                       1
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Razão do final: Bloqueio
ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ
Conclusão:

Total de estados globais obtidos: 8
Tempo de verificação: 0 horas, 0 minutos e 18 segundos
Especificação com bloqueio
```

c) As Seqüências de Transições da Árvore de Alcançabilidade

```
Estado raiz = (1)
 (1) -Modulo 4 -Transição TRANSMIT1[0,0] -Prob 1.00000 --> (2)
 (2) -Modulo 3 -Transição TIME2 -Prob 1.00000 --> (3)
 (3) -Modulo 4 -Transição TRANSMIT2[500,500] -Prob 1.00000 --> (4)
 (4) -Modulo 1 -Transicão NETWORK1 -Prob 0.00100 --> (5)
 (5) -Modulo 2 -Transição RECEBE1 -Prob 1.00000 --> (6)
 (6) -Modulo 3 -Transição TIME5[4500,4500] -Prob 1.00000 --> (7)
 (7) -Modulo 4 -Transição TRANSMIT3 -Prob 1.00000 --> (8)
** bloqueio **
```

B.6 O exemplo "Abracadabra"

a) A Especificação em Estelle do Sistema

```
specification ABRACADABRA systemactivity;
  default individual queue; timescale seconds;

(*************************************************)
  const
    MTD = 1;     (* Tempo de transito medio *)
    EPSILON = 1;
    N = 3;       (* Numero maximo de tentativas *)

{O Avaliador nao permite a utilizacao de variaveis na clausula DELAY. Deste
modo, definiu-se o tempo de retransmissao 'P' como uma constante.        }
    P = 4; {2 * (MTD + EPSILON);}
```

```
(*********************************************************)
 type

  data_type =  ARRAY[0..4] OF CHAR ;
  pdu_type = RECORD
           Sequencia   : CHAR    ; {Variavel E e R do protocolo}
           SDU         : data_type ;
           CodigoPrt   : ARRAY[0..1] OF CHAR ; {Tipo da Informacao}
         END ;

(*********************************************************)
 channel ABRA_interface(user,provider);
  by user :
   ConReq;ConResp;DatReq(SDU:data_type);DisReq;
  by provider :
   ConInd;ConConf;DatInd(SDU:data_type);DisInd;

 channel MEDIUM_interface(user,provider);
  by user :
   UnitReq(PDU:pdu_type);
  by provider :
   UnitInd(PDU:pdu_type);

(*********************************************************)
 module ABRA_ENTITY_TYPE activity;
  ip UCEP   : ABRA_interface(provider);
    MCEP    : MEDIUM_interface(user);
    export H : integer;
 end;

(*********************************************************)
 body ABRA_ENTITY_BODY for ABRA_ENTITY_TYPE;

  type code_type = (CR,CC,DR,DC,DT,AK,SS) ;

{ O codigo SS indica uma mensagem Sem Sentido }

  var
   N_attempts : integer; (* contador de tentativas *)
   E : integer;        (* Numero de sequencia de emissao *)
   R : integer;        (* Numero de sequencia de recepcao *)
   First_Received_PDU : boolean;
   Emission_Buffer : pdu_type;

   probability : real;

   {$Verificar var N_attempts}
   {$Verificar var E}
   {$Verificar var R}
   {$Verificar var First_Received_PDU}
   {$Verificar var probability}

{Estas variaveis retornam os PDU's das rotinas externas              }

  state
   CLOSED,WFCC (* Espera por CC *),OPEN,WFAK (* Espera por AK *),
   CLOSING,ERROR,WFUR (* Espera por resposta do usuario *);

 VAR
  pdu_aux  : pdu_type ;
```

```
data_aux : data_type ;

PROCEDURE BuildCR(VAR PDU:pdu_Type) ;
(*Prepara um pedido de conexao*)
 BEGIN
  PDU.CodigoPrt := 'CR' ;
 END ;

PROCEDURE BuildCC(VAR PDU:pdu_type) ;
(*Prepara confirmacao de conexao*)
 BEGIN
  PDU.CodigoPrt := 'CC' ;
 END ;

PROCEDURE BuildDR(VAR PDU:pdu_type) ;
(*Prepara pedido de desconexao*)
 BEGIN
  PDU.CodigoPrt := 'DR' ;
 END ;

PROCEDURE BuildDC(VAR PDU:pdu_type) ;
(*Prepara confirmacao de desconexao*)
 BEGIN
  PDU.CodigoPrt := 'DC' ;
 END ;

PROCEDURE BuildDT(UDATA:data_type;N:INTEGER;VAR PDU:pdu_type) ;
(*Prepara dado a ser transmitiodo*)
 BEGIN
  PDU.CodigoPrt :='DT' ;
  PDU.SDU := UDATA ;
  IF N = 1 THEN PDU.Sequencia := '1' ELSE PDU.Sequencia := '0' ;
 END ;

PROCEDURE BuildAK(N:INTEGER;VAR PDU:pdu_type) ;
(*Prepara confirmacao de dado*)
 BEGIN
  PDU.CodigoPrt := 'AK' ;
  IF N = 1 THEN PDU.Sequencia := '1' ELSE PDU.Sequencia := '0' ;
 END ;

FUNCTION Code(PDU:pdu_type): code_type ;
(*Indica o codigo da mensagem que chegou*)

 BEGIN

  IF (PDU.CodigoPrt[0] ='C') AND (PDU.CodigoPrt[1] = 'C') THEN
   Code := CC
    ELSE
   IF (PDU.CodigoPrt[0] ='C') AND (PDU.CodigoPrt[1] = 'R') THEN
    Code := CR
     ELSE
   IF (PDU.CodigoPrt[0] ='D') AND (PDU.CodigoPrt[1] = 'R') THEN
    Code := DR
     ELSE
   IF (PDU.CodigoPrt[0] ='D') AND (PDU.CodigoPrt[1] = 'C') THEN
     Code := DC
      ELSE
   IF (PDU.CodigoPrt[0] ='D') AND (PDU.CodigoPrt[1] = 'T') THEN
      Code := DT
```

```
      ELSE
    IF (PDU.CodigoPrt[0] ='A') AND (PDU.CodigoPrt[1] = 'K') THEN
     Code := AK
    ELSE
      Code := SS ;
   END ;

  FUNCTION Seq(PDU:pdu_type) : INTEGER ;
  (*Indica a sequencia do PDU que chegou*)
   BEGIN
    IF PDU.Sequencia = '0' THEN Seq := 0
     ELSE
      IF PDU.Sequencia = '1' THEN Seq := 1
       ELSE
        Seq := 2 ;
   END ;

  PROCEDURE Data(PDU:pdu_type;VAR Data:data_type) ;
  (*Retira do PDU o dado transmitido*)
   BEGIN
    Data := PDU.SDU ;
   END ;

{ Este procedimento foi incluido para inicializar as variaveis do protocolo,
em situacoes nao previstas na especificacao original, mas que parecem ser ne-
cessarias }

  PROCEDURE Inicializa;
   (*Inicializa as variaveis do protocolo*)
   BEGIN
     N_attempts := 0; E := 0; R := 0;
     First_Received_PDU := true
   END;

  initialize (* de ABRA_ENTITY_BODY *)
   to CLOSED
   begin
     N_attempts := 0; E := 0; R := 0;
     First_Received_PDU := true
   end;

  (* Fase de abertura de conexao *)
  trans
   (* comportamento do iniciador da conexao *)
   from CLOSED to WFCC
     when UCEP.Conreq
     name AB1:
     begin probability := 0.5;
       BuildCR(pdu_aux);
       output MCEP.UnitReq(pdu_aux);
       H := 3
     end;
   from WFCC
     when MCEP.UnitInd
       provided (Code(PDU) = CC) or (Code(PDU) = CR)
         to OPEN
         name AB2:
         begin probability := 0.5;
           H := 0;
           output UCEP.ConConf;
```

```
(**)        N_attempts := 0;
          end;
        provided Code(PDU) = DR
          to CLOSED
          name AB3:
          begin probability := 0.5;
            H := 10;
            output UCEP.DisInd;
            BuildDC(pdu_aux) ;
            output MCEP.UnitReq(pdu_aux);
(**)        N_attempts := 0;
          end;
        provided otherwise
          to WFCC
          name AB4:
          begin probability := 0.5;
            H := 10
          end;
      from WFCC to CLOSING
        when UCEP.DisReq
          name AB5:
          begin probability := 0.5;
            H := 10;
            BuildDR(pdu_aux) ;
            output MCEP.UnitReq(pdu_aux);
(**)        N_attempts := 0;
        end;
      from WFCC
        provided N_attempts < N
          to WFCC
          delay(P)
          name AB6:
          begin probability := 0.5;
            H := 10;
            BuildCR(pdu_aux) ;
            output MCEP.UnitReq(pdu_aux);
            N_attempts := N_attempts + 1
          end;
        provided otherwise
          to ERROR
          name AB7:
          begin probability := 0.5;
            H := 10
          end;

      (* comportamento do respondedor da conexao *)
      from CLOSED
        when MCEP.UnitInd
          provided Code(PDU) = CR
            to WFUR
            name AB8:
            begin probability := 0.5;
              H := 3;
              output UCEP.ConInd
            end;
          provided Code(PDU) = DR
            to CLOSED
            name AB9:
            begin probability := 0.5;
              H := 10;
```

```
        BuildDC(pdu_aux) ;
        output MCEP.UnitReq(pdu_aux)
       end;
      provided otherwise
       to CLOSED
       name AB10:
       begin probability := 0.5;
        H := 10;
       end;
   from WFUR
    when UCEP.ConResp
     to OPEN
     name AB11:
     begin probability := 0.5;
      H := 0;
      BuildCC(pdu_aux) ;
      output MCEP.UnitReq(pdu_aux)
     end;
    when UCEP.DisReq
     to CLOSING
     name AB12:
     begin probability := 0.5;
      H := 10;
      BuildDR(pdu_aux) ;
      output MCEP.UnitReq(pdu_aux) ;
     end;

  (* Fase de transferencia de dados *)
  trans
   from OPEN to WFAK
    (* pedido de transferencia de dados *)
    when UCEP.DatReq
     name AB13:
     begin probability := 0.5;
      H := 3;
      BuildDT(SDU,E,pdu_aux) ;
      Emission_buffer := pdu_aux ;
      output MCEP.UnitReq(Emission_buffer);

{Uma vez que se espera receber  o valor da variavel de sequencia 'E' comple-
mentado, parece ser necessario complementar o valor desta  variavel  apos a
transmissao.   }
       E := (E + 1) mod 2;
       N_attempts := 0
      end;
   from WFAK
    (* recepcao de um AK *)
    when MCEP.UnitInd
     (* com numero de sequencia correto *)
{Alterou-se a variavel de sequencia para E.}
      provided (Code(PDU) = AK) AND (Seq(PDU) = E)
       to OPEN
       name AB14:
       begin probability := 0.5;
        H := 3;
{Retirou-se o complemento da variavel E                          }
        First_Received_PDU := false
       end;
{Alterou-se a variavel de sequencia para E.}
      provided (Code(PDU) = AK) AND (Seq(PDU) <> E)
```

```
      to ERROR
      name AB15:
      begin probability := 0.5;
       H := 10
      end;
 from WFAK
  provided N_attempts < N
   to WFAK
   delay(P)
   name AB16:
   begin probability := 0.5; (* Retransmissao da PDU DT *)
    H := 10;
    output MCEP.UnitReq(Emission_Buffer);
    N_attempts := N_attempts + 1
   end;
  provided otherwise
   to ERROR
   name AB17:
   begin probability := 0.5;
    H := 10
   end;
from WFAK, OPEN
 when MCEP.UnitInd
  provided (Code(PDU) = DT) and (Seq(PDU) = R)
   to same
   name AB18:
   begin probability := 0.5;
    H := 3;
    Data(PDU,data_aux) ;
    output UCEP.DatInd(data_aux);
    R := (R + 1) mod 2;
    BuildAK(R,pdu_aux) ;
    output MCEP.UnitReq(pdu_aux);
    First_Received_PDU := false
   end;
  provided (Code(PDU) = DT) and (Seq(PDU) <> R)
   to same
   name AB19:
   begin probability := 0.5;
    H := 10;
    BuildAK(R,pdu_aux) ;
    output MCEP.UnitReq(pdu_aux);
    First_Received_PDU := false
   end;
  provided (Code(PDU) = CR) and First_Received_PDU
   to same
   name AB20:
   begin probability := 0.5;
    H := 10;
    BuildCC(pdu_aux) ;
    output MCEP.UnitReq(pdu_aux);
   end;
  provided Code(PDU) = DR
   to CLOSED
   name AB21:
   begin probability := 0.5;
    H := 0;
    output UCEP.DisInd;
    BuildDC(pdu_aux) ;
    output MCEP.UnitReq(pdu_aux);
```

```
          First_Received_PDU := true;
(**)      Inicializa;
         end;
        provided otherwise
         to ERROR
         name AB22:
         begin probability := 0.5;
          H := 10
         end;

   (* Fase de fechamento da conexao *)
   trans
    from OPEN, WFAK
     when UCEP.DisReq
     to CLOSING
      name AB23:
      begin probability := 0.5;
       H := 3;
       N_attempts := 0;
       BuildDR(pdu_aux) ;
       output MCEP.UnitReq(pdu_aux);
      end;
    from CLOSING
     when MCEP.UnitInd
      provided (Code(PDU) = DC) or (Code(PDU) = DR)
       to CLOSED
       name AB24:
       begin probability := 0.5;
        H := 0;
(**)    Inicializa;
       end;
      provided otherwise
       to CLOSING
       name AB25:
       begin probability := 0.5;
        H := 10
       end;
    from CLOSING
     provided N_attempts < N
      to CLOSING
      delay(P)
      name AB26:
      begin probability := 0.5; (* Retransmissao da PDU DR *)
       H := 10;
       BuildDR(pdu_aux) ;
       output MCEP.UnitReq(pdu_aux);
       N_attempts := N_attempts + 1
      end;
     provided otherwise
      to CLOSED
      name AB27:
      begin probability := 0.5;
       H := 10;
(**)    Inicializa;
      end;

   (* Fase de erro *)
   trans
    from ERROR to CLOSING
    name AB28 :
```

```
    begin probability := 0.5;
     H := 10;
     output UCEP.DisInd;
     N_attempts := 0;
     BuildDR(pdu_aux) ;
     output MCEP.UnitReq(pdu_aux);
    end;

  end; (* de ABRA_ENTITY_BODY *)

(*********************************************************)
 module MEDIUM_TYPE activity;
  ip MCEP : array[0..1] of MEDIUM_interface(provider);
 end;

(*********************************************************)
 body MEDIUM_BODY for MEDIUM_TYPE;

  var probability : real;

  {$Verificar var probability}

  initialize (* de MEDIUM_BODY *)
   name INIC_MEIO:
   begin
   end;

  trans
  (* comportamento do meio de comunicacao *)
   when MCEP[0].UnitReq
    name MEIO01:
    begin probability := 0.5;
     output MCEP[1].UnitInd (PDU);
    end;
   when MCEP[1].UnitReq
    name MEIO10:
    begin probability := 0.5;
     output MCEP[0].UnitInd (PDU);
    end;

  end; (* de MEDIUM_BODY *)

(*********************************************************)
 module USER_ENTITY_TYPE activity (AbreCon : BOOLEAN);
  ip UCEP : ABRA_interface (user);
 end;

(*********************************************************)
 body USER_ENTITY_BODY for USER_ENTITY_TYPE;

 var
   Informacao        : data_type;
   probability        : real;

   {$Verificar var Informacao}
   {$Verificar var probability}
   {$Afetar var := Informacao}

 initialize
  begin
```

```
    end;

  trans
    begin output ucep.conreq;
    end;

    when ucep.conind
    begin end;

    begin output ucep.conresp;
    end;

    when ucep.conconf
    begin end;

    begin output ucep.datreq( informacao );
    end;

    when ucep.datind
    begin end;

    begin output ucep.disreq;
    end;

    when ucep.disind
    begin end;

  end; (* de USER_ENTITY_BODY *)

(*********************************************************)
  modvar
    UA,UB : USER_ENTITY_TYPE;
    A,B : ABRA_ENTITY_TYPE;
    DG : MEDIUM_TYPE;

(*********************************************************)
  initialize (* de ABRACADABRA *)
    name init_S_ENT :
    begin
      init DG with MEDIUM_BODY;
      init A with ABRA_ENTITY_BODY;
      init B with ABRA_ENTITY_BODY;
      init UA with USER_ENTITY_BODY (TRUE);
      init UB with USER_ENTITY_BODY (TRUE);
      connect A.MCEP to DG.MCEP[0];
      connect B.MCEP to DG.MCEP[1];
      connect UA.UCEP to A.UCEP;
      connect UB.UCEP to B.UCEP
    end;

end. (* de ABRACADABRA *)
```

b.1) A Descrição das Propriedades do Sistema com Bloqueio

Especificação: ABRACADABRA
íí
Expressões não usadas: 10
6
5
íí
Razão do final: Bloqueio
íí
Conclusão:

Total de estados globais obtidos: 60

Tempo de Geração: 0 horas, 2 minutos e 21 segundos

Especificação com bloqueio

b.2) A Descrição das Propriedades do Sistema sem Bloqueio

Especificação: ABRACADABRA
íí
Expressões não usadas: 8
6
5
íí
Razão do final: Aborto pelo usu rio
íí
Conclusão parcial:

Total de estados globais obtidos: 549

Tempo de Geração: 9 horas, 27 minutos e 59 segundos

Especificação não limitada

c) As Seqüências de Transições do Sistema com Bloqueio

Estado raiz = (1)
(1) -Modulo 4 -Transiçao USER1[0,0] -Prob 0.50000 --> (2)
(2) -Modulo 5 -Transiçao USER1[0,0] -Prob 0.50000 --> (3)
(3) -Modulo 2 -Transiçao AB1 -Prob 0.50000 --> (4)
(4) -Modulo 3 -Transiçao AB1 -Prob 0.50000 --> (5)
(5) -Modulo 2 -Transiçao AB6[4,4] -Prob 0.50000 --> (6)
(6) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (7)
(7) -Modulo 1 -Transiçao MEIO01 -Prob 0.50000 --> (8)
(8) -Modulo 3 -Transiçao AB2 -Prob 0.50000 --> (9)
(9) -Modulo 2 -Transiçao AB2 -Prob 0.50000 --> (10)
(10) -Modulo 1 -Transiçao MEIO01 -Prob 0.50000 --> (11)
(11) -Modulo 4 -Transiçao USER4 -Prob 0.50000 --> (12)
(12) -Modulo 3 -Transiçao AB20 -Prob 0.50000 --> (12)
** BACKTRACK TO (12) **
(12) -Modulo 5 -Transiçao USER4 -Prob 0.50000 --> (13)
(13) -Modulo 5 -Transiçao USER8[0,0] -Prob 0.50000 --> (14)

(14) -Modulo 3 -Transiçao AB23 -Prob 0.50000 --> (15)
(15) -Modulo 4 -Transiçao USER8[0,0] -Prob 0.50000 --> (16)
(16) -Modulo 5 -Transiçao USER1[0,0] -Prob 0.50000 --> (17)
(17) -Modulo 3 -Transiçao AB26[4,4] -Prob 0.50000 --> (18)
(18) -Modulo 2 -Transiçao AB23 -Prob 0.50000 --> (19)
(19) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (20)
(20) -Modulo 1 -Transiçao MEIO01 -Prob 0.50000 --> (21)
(21) -Modulo 2 -Transiçao AB24 -Prob 0.50000 --> (22)
(22) -Modulo 4 -Transiçao USER1[0,0] -Prob 0.50000 --> (23)
(23) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (24)
(24) -Modulo 3 -Transiçao AB25 -Prob 0.50000 --> (25)
(25) -Modulo 3 -Transiçao AB24 -Prob 0.50000 --> (26)
(26) -Modulo 3 -Transiçao AB1 -Prob 0.50000 --> (27)
(27) -Modulo 2 -Transiçao AB1 -Prob 0.50000 --> (5)
** BACKTRACK TO (27) **
(27) -Modulo 3 -Transiçao AB6[4,4] -Prob 0.50000 --> (28)
(28) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (29)
(29) -Modulo 2 -Transiçao AB1 -Prob 0.50000 --> (30)
(30) -Modulo 2 -Transiçao AB6[4,4] -Prob 0.50000 --> (31)
(31) -Modulo 1 -Transiçao MEIO01 -Prob 0.50000 --> (32)
(32) -Modulo 3 -Transiçao AB6[0,0] -Prob 0.50000 --> (33)
(33) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (34)
(34) -Modulo 1 -Transiçao MEIO01 -Prob 0.50000 --> (35)
(35) -Modulo 2 -Transiçao AB6[4,4] -Prob 0.50000 --> (36)
(36) -Modulo 2 -Transiçao AB3 -Prob 0.50000 --> (37)
(37) -Modulo 1 -Transiçao MEIO01 -Prob 0.50000 --> (38)
(38) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (39)
(39) -Modulo 4 -Transiçao USER3 -Prob 0.50000 --> (40)
(40) -Modulo 3 -Transiçao AB6[0,0] -Prob 0.50000 --> (41)
(41) -Modulo 3 -Transiçao AB7[0,0] -Prob 0.50000 --> (42)
(42) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (43)
(43) -Modulo 1 -Transiçao MEIO01 -Prob 0.50000 --> (44)
(44) -Modulo 4 -Transiçao USER1[0,0] -Prob 0.50000 --> (45)
(45) -Modulo 3 -Transiçao AB28[0,0] -Prob 0.50000 --> (46)
(46) -Modulo 5 -Transiçao USER3 -Prob 0.50000 --> (47)
(47) -Modulo 3 -Transiçao AB26[4,4] -Prob 0.50000 --> (48)
(48) -Modulo 2 -Transiçao AB8 -Prob 0.50000 --> (49)
(49) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (50)
(50) -Modulo 3 -Transiçao AB25 -Prob 0.50000 --> (50)
** BACKTRACK TO(50) **
(50) -Modulo 5 -Transiçao USER1[0,0] -Prob 0.50000 --> (51)
(51) -Modulo 3 -Transiçao AB26[4,4] -Prob 0.50000 --> (52)
(52) -Modulo 3 -Transiçao AB25 -Prob 0.50000 --> (53)
(53) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (54)
(54) -Modulo 3 -Transiçao AB26[4,4] -Prob 0.50000 --> (55)
(55) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (56)
(56) -Modulo 3 -Transiçao AB25 -Prob 0.50000 --> (57)
(57) -Modulo 3 -Transiçao AB27[0,0] -Prob 0.50000 --> (58)
(58) -Modulo 1 -Transiçao MEIO10 -Prob 0.50000 --> (59)
(59) -Modulo 3 -Transiçao AB8 -Prob 0.50000 --> (60)
** bloqueio **

B.7 O exemplo "SomaDif"

a) A Descrição das Propriedades do Sistema

Especificação: EXEMPLO
Expressões não usadas: 1
Razão do final: Geração completada
Conclusão:

Total de estados globais obtidos: 9

Tempo de Geração: 0 horas, 0 minutos e 2 segundos

Especificação limitada e reinicializável

Estados globais com soma das probabilidades de saída diferente de 1:
9,8,7,6,5,4,3,2,1