Avaliação de Desempenho de Sistemas Distribuídos Especificados em Estelle

Pedro Mejía Ochoa
COPPE/UFRJ e HONDUTEL
Tegulcigalpa, Honduras
pmejia@huracan.cr

Edmundo de Souza e Silva
UFRJ, NCE e Dept. de Computação
Cx.P. 2324, CEP 20001, Rio de Janeiro
edmundo@ufrj.bitnet

Aloysio de Castro Pinto Pedroza
UFRJ, COPPE/Elétrica e Dept. de Eletrônica/EE
Cx.P. 68504, CEP 20001, Rio de Janeiro

Sumário

Estelle é uma linguagem para a especificação formal de sistemas distribuídos padronizada pela ISO. O objetivo principal das técnicas de especificação formal é o de dar suporte para que protocolos desenvolvidos para sistemas distribuídos sejam descritos sem ambigüidades. É parte integrante de um projeto de protocolos (e sistemas distribuídos em geral) a criação de modelos matemáticos que possibilitem a análise de desempenho do sistema. Para minimizar o esforço de modelagem, é desejável que, a partir de uma mesma especificação (formal) do sistema, medidas de desempenho possam também ser obtidas, além das outras análises normalmente feitas para evitar erros de projeto. A contribuição deste trabalho consiste no desenvolvimento de uma ferramenta capaz de gerar automaticamente, em forma simbólica, a cadeia de Markov associada a um sistema a partir da especificação em Estelle, para posterior cálculo das medidas de desempenho desejadas. Este trabalho integra duas ferramentas previamente desenvolvidas: um compilador Estelle que gera uma forma intermediária de Estelle com estruturas de dados de fácil manejo e uma ferramenta de modelagem de desempenho que gera o modelo matemático de sistemas a partir de uma especificação orientada a objetos.

Abstract

Estelle is a formal specification language, ISO standard, for distributed systems. The main goal of formal specification techniques is to provide the necessary support for the development of protocols for distributed systems without errors or ambiguities. However, it is also part of the design process of protocols (or distributed systems in general) to develop models to evaluate the performance of the system being designed. In order to minimize the modeling effort it is desirable that, from the same (formal) system specification, the designer can also obtain the performance measures of interest. The contribution of this work is to develop a tool that can automatically generate, in symbolic form, the analytical performance model (the Markov chain) from a system specification given in Estelle. The work integrates two previously developed tools: an Estelle compiler that generates an intermediate form with simple data structures and a performance modeling tool that generates the matematical model of computer/communication systems from an object oriented high level specification.

1 Introdução

A avaliação de desempenho de sistemas de computação, em particular sistemas distribuídos e protocolos de comunicação, é muito importante para o projeto, pois permite prever o comportamento de tais sistemas durante o seu funcionamento. A avaliação em geral é feita usando-se modelos analíticos que são abstrações matemáticas do sistema real, e modelos Markovianos tem sido uma das principais ferramentas disponíveis para o analista. Entretanto, devido a complexidade dos sistemas (e.g. protocolos de comunicação) a tarefa de encontrar uma cadeia de Markov que representa o sistema torna-se impraticável, a menos que o analista disponha de ferramentas que o auxiliem neste trabalho.

Um paradigma muito usado para especificação e geração da cadeia de Markov que representa um sistema é a modelagem por redes de Petri estocásticas [20], e existem vários exemplos da utilização deste paradigma para análise de protocolos de comunicação de dados [30, 19, 12]. As redes de Petri estocáticas tem a vantagem de possuir uma representaçã gráfica natural. Entretanto, devido a simplicidade das primitivas básicas de modelagem (lugares, transições e fichas), redes de Petri não são convenientes para especificar sistemas complexos, pois exigem um grande esforço do analista para representar o sistema em termos destas primitivas básicas.

Com o objetivo de construir uma ferramenta de modelagem que pudesse ser suficientemente geral para ser usada em diferentes domínios de aplicação e ainda fornecer uma interface simples e poderosa para que interfaces de mais alto nível pudessem ser construídas e talhadas para uma determinada aplicação, Berson *et al* [1] propuseram um novo paradigma de modelagem. Neste paradigma, sistemas são definidos em termos de instâncias de objetos (componentes do sistema) que interagem pela troca de mensagens. Características de diferentes aplicações são representadas por tipos distintos de objetos usados na especificação de modelos. A partir da especificação, a cadeia de Markov que modela o sistema é gerada em forma simbólica.

Linguagens formais de especificação de protocolos tem sido desenvolvidas e recentemente algumas foram padronizadas, como por exemplo, a linguagem Estelle [13, 14]. Estelle é uma linguagem de especificação formal de sistemas distribuídos em geral, definida pela ISO (International Organization for Standardization) e desenvolvida em particular para a especificação de protocolos de comunicação de dados. Uma especificação Estelle é formada pela descrição de estruturas de máquinas de estados finitas que se comunicam entre si e as suas ações que são definidas com cláusulas Pascal ISO.

Para minimizar o esforço de modelagem, é desejável que, a partir de uma mesma especificação (formal) de um protocolo, medidas de desempenho possam também ser obtidas, além das outras análises normalmente feitas para evitar erros de projeto (e.g., deteção de entraves, geração de código). Nosso objetivo consiste no desenvolvimento de uma ferramenta capaz de gerar a cadeia de Markov associada a um modelo de protocolo de comunicação (ou um sistema distribuído em geral) a partir da sua especificação em Estelle, e posteriormente calcular as medidas de desempenho desejadas. Este trabalho integra duas ferramentas previamente desenvolvidas: um compilador ESTELLE que gera uma forma intermediária de Estelle com estruturas de dados de fácil manejo [6] e a ferramenta para especifição de sistemas de computação desenvolvida em [10, 9] mencionada acima. Sendo assim, as características de Estelle são automaticamente mapeadas no modelo orientado a objeto de [1], para posterior análise de desempenho.

Outros trabalhos relacionados à análise automática de desempenho de protocolos de comunicação podem ser encontrados em [17, 25, 26, 11]. Estes trabalhos entretanto, visam o desenvolvimento de métodos de cálculo de medidas de desempenho a partir dos estados mar-

kovianos do sistema. Eles pressupõem a existência de uma técnica para a descrição formal do sistema, associada a uma ferramenta que gere os estados da cadeia de Markov.

Este artigo está organizado da seguinte forma : na segunda seção são apresentadas as principais características de Estelle, assim como as restrições impostas neste trabalho. A metodologia orientada a objeto utilizada [1] é também brevemente apresentada. Na terceira seção, a ferramenta desenvolvida para a análise de desempenho de sistemas distribuídos especificados em Estelle é descrita. Na quarta seção é abordado sucintamente o problema de explosão do espaço de estados. Na quinta seção são apresentados exemplos que demostram a utilidade da ferramenta e na sexta seção as conclusões do trabalho são apresentadas.

2 Conceitos Básicos

Nesta seção serão descritas as principais características da linguagem de especificação de sistemas distribuídos **Estelle**, e será apresentada uma metodologia orientada a objeto para especificação e geração de modelos Markovianos. Estes temas são apresentados de forma introdutória. Maiores detalhes podem ser encontrados em [15, 14, 2, 13, 4, 3, 22, 23, 5, 28] para Estelle e a [10, 1, 9] para o modelo orientado a objeto.

2.1 Estelle

Estelle é uma linguagem de descrição formal padronizada pela ISO, apropriada para a especificação formal de sistemas distribuídos e, em particular, de protocolos de comunicação. Uma especificação em Estelle é constituída por uma hierarquia de módulos cooperantes. Cada módulo é definido como um tipo; instâncias de módulos podem ser criadas, destruídas, conectadas e desconectadas com o uso de um conjunto de primitivas apropriadas, definidas por esta linguagem.

Os módulos interconectam-se através de pontos de interação, aos quais são associados filas FIFO que recebem as interações trocadas entre módulos. A comunicação por meio de variáveis compartilhadas é permitida entre módulos de níveis hierárquicos adjacentes, isto é, um módulo filho pode exportar variáveis que são acessíveis ao módulo que o criou (módulo pai). O comportamento de cada módulo é descrito por um conjunto de sistemas de transições do tipo predicado-ação e o tratamento de dados é especificado com o auxílio da linguagem Pascal.

A execução de uma especificação em Estelle é determinada por atributos associados a cada módulo que a constitui. No nível mais alto da hierarquia encontram-se módulos com atributo *Systemprocess* e *Systemactivity* que executam de forma paralela assíncrona. Os refinamentos de módulos com atributos *Systemprocess* ou *Process* executam de forma paralela assíncrona, segundo uma semântica própria da linguagem Estelle. Os refinamentos de módulos com atributos *Systemactivity* e *Activity* executam de forma não determinística segundo a semântica das redes de Petri, modelo de base bastante difundido e aceito para a especificação de sistemas concorrentes.

Neste trabalho consideramos especificações que possuem o atributo *Systemactivity* e que são refinados segundo módulos que recebem o atributo *Activity*. Esta abordagem segue uma interessante proposta para Estelle apresentada em [3].

Um exemplo simples de Estelle é dado a seguir. Este exemplo será utilizado no restante deste trabalho.

> 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, caso contrário ele nada faz.

A especificação Estelle deste sistema é dada a seguir, onde a mensagem MSG representa envio da informação produzida e a mensagem LIB indica que o consumidor está pronto p receber uma mensagem. A estrutura desta especificação é mostrada na figura 2.1.

```
Specification procon systemactivity;
default individual queue;timescale seconds;

channel
    interface_type (i_pro,i_con);
    by i_pro:
      MSG;
    by i_con:
      LIB;

module pro_type activity;
    ip p1: interface_type (i_pro);
end;

body pro_body for pro_type;
    const
      pros = 100;
    state processing, wflib;
    initialize to wflib
      begin
      end;
    trans
    from wflib to processing
      when p1.LIB
        begin
        end;
    from processing to wflib
      delay(pros)
        begin
          output p1.MSG;
        end;
end;

module con_type activity;
    ip p2: interface_type (i_con);
end;

body con_body for con_type;
    var
      flag : boolean;
    initialize
      begin
        flag := true;
      end;
    trans
      provided flag = true
        begin
          output p2.LIB;
          flag := false ;
        end;
    trans
      when p2.MSG
        begin
          output p2.lib
        end;
end;

modvar pro:pro_type;
    con:con_type;

initialize
```

```
    begin
      init pro with pro_body;
      init con with con_body;
      connect pro.p1 to con.p2
    end;
end.
```

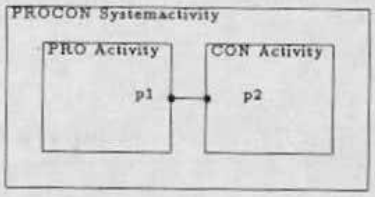

Figura 2.1: Estrutura da especificação produtor-consumidor

2.2 A Metodologia Orientada a Objeto

Um sistema modelado pelo paradigma de Berson *et al* [1] orientado a objeto é composto por um conjunto de componentes chamados de objetos que interagem entre si pela troca de mensagens. Cada objeto possui um estado interno que muda ao longo do tempo a partir de eventos gerados internamente e ao receber mensagens de outros objetos. O estado de um objeto determina os eventos que ele pode gerar e as taxas com que estes eventos ocorrem.

Um evento pode causar não somente a troca do estado do objeto que o gerou, mas também do estado de outros objetos, pelo envio de mensagens geradas pelo evento. A especificação de um objeto inclui a definição de como o objeto reage com o recebimento de mensagens.

O estado global do sistema é formado pelo conjunto de estados internos dos objetos e a lista de mensagens não entregues. As mensagem são entregues em tempo zero e é nulo o tempo de reação do objeto a uma mensagem recebida. Por conseguinte, existem estados transitórios ou evanescentes. Estes estados são aqueles com uma ou mais mensagens não entregues.

Um objeto é definido formalmente da seguinte forma :

$$\varphi \stackrel{\Delta}{=} (I, S, S_0, \mathcal{E}, M^r, M^s, P, \delta', \delta)$$

onde: (a) $[I :]$ Nome do objeto; (b) $[S :]$ Conjunto de estados do objeto; (c) $[S_0 \in S :]$ Estado inicial do objeto; (d) $[\mathcal{E} :]$ Conjunto de eventos que o objeto pode gerar; (e) $[M^r :]$ Conjunto de mensagens que o objeto pode receber; (f) $[M^s :]$ Conjunto de mensagens que o objeto pode enviar; (g) $[P :]$ Função taxa do objeto: $P : S \times \mathcal{E} \rightarrow +\Re$; (h) $[\delta']$ Função de eventos do objeto: $\delta' : S \times \mathcal{E} \rightarrow 2^{(0,1] \times S \times [M^s]}$ $[M^s] \in M^s$ é uma lista ordenada de mensagens; (i) $[\delta]$ Função de mensagens do objeto: $\delta : S \times M^r \rightarrow 2^{(0,1] \times S \times [M^s]}$.

Na definição formal do objeto dada acima, δ é uma função tal que, dado um estado e um evento, retorna o conjunto de possíveis respostas ao evento, a lista de mensagens a serem enviadas e a probabilidade desta resposta ocorrer. Associada a cada evento existe a função de taxa do objeto, que retorna a taxa do evento ocorrer.

O estado $\mathcal{S}$ de um sistema χ modelado com este paradigma consiste num vetor de estados dos objetos (um estado para cada objeto) e uma lista de mensagens:

$$\mathcal{S} = \{< s_1, s_2, \ldots, s_N >, [m_1, \ldots, m_n]\}$$

onde N é o número de objetos do modelo, $s_i \in S$.
Seja a função γ definida em termos das funções δ e δ':

$$\gamma \ : \ (< s_1, \ldots, s_i, \ldots, s_N >, [m_1, \ldots, m_k]) \\ \rightarrow (< s_1, \ldots, s_i', \ldots, s_N >, [m_2, \ldots, m_k, m_{k+1}, \ldots, m_{k+m}])$$

se a mensagem m_1 tem como destinatário o objeto φ_i, e $m_{k+1}, \ldots, m_{k+n}$ são mensagens geradas pelo objeto φ_i como resposta da mensagem m_1, ou seja: $(p, s_i', [m_{k+1}, \ldots, m_{k+n}]) \in \delta_i(s_i, m_1), p > 0$.

$$\gamma \ : \ (< s_1, \ldots, s_i, \ldots, s_N >, [\,]) \\ \rightarrow (< s_1, \ldots, s_i', \ldots, s_N >, [m_1, \ldots, m_k])$$

se e é um evento que o objeto φ_i gerou, e $(p, s_i', [m_1, \ldots, m_k]) \in \delta_i'(s_i, e)$.

Um estado é chamado de alcançável se existe uma seqüência válida do estado inicial $\mathcal{S}_0$ a ele. Um estado alcançável sem mensagens pendentes a serem entregues $(< s_1, \ldots, s_N >, [\,])$ é chamado de tangível. De outra forma, o estado é chamado de evanescente. A cadeia de Markov gerada possui apenas os estados tangíveis.

Baseada nesta descrição formal do paradigma orientado a objeto existem ferramentas que geram a cadeia de Markov a partir de uma especificação de sistemas neste paradigma [10, 16]. Abaixo será descrita sucintamente a arquitetura básica dessas ferramentas. Neste trabalho, utilizaremos a implementação de [10].

A ferramenta de especifição e análise de sistemas de computação que usa o paradigma orientado a objeto, é organizada em quatro níveis:
(1) O **núcleo**, que aceita a descrição do sistema em termos de objetos, eventos e mensagens. A partir desta descrição e de um estado inicial, o núcleo gera a matriz de transição de estados (a cadeia de Markov) do sistema.
(2) O nível de **definição do tipo de objeto**, onde diferentes tipos de objetos são especificados. Objetos com o mesmo comportamento, diferindo apenas no valor dos parâmetros especificados para cada um são do mesmo tipo.
(3) No nível de **aplicação**, o usuário define um modelo criando objetos utilizando-se da biblioteca de tipos de objetos definidos no nível inferior e especificando os parâmetros necessários.
(4) O nível de **interface** permite que o usuário final se abstraia dos níveis anteriores, oferecendo uma linguagem de alto nível para a especificação do modelo.

O núcleo está implementado em Prolog e a ferramenta deste trabalho (para avaliação de desempenho de protocolos especificados em Estelle) faz interface com este núcleo, como será detalhado na seção seguinte. Algumas modificações no núcleo foram necessárias para permitir a especificação de prioridade de módulos e transições. Outras modificações foram feitas com relação a maneira pela qual os estados são explorados, para implementar o algoritmo de redução de estados que será comentado na seção 4.

3 A Ferramenta para Análise de Desempenho de Protocolos Especificados em Estelle

Nesta seção serão descritas as principais características da ferramenta desenvolvida que, a partir de uma especificação Estelle, gera a cadeia de Markov correspondente ao sistema especificado. Inicialmente é apresentada a arquitetura da ferramenta desenvolvida. Em seguida é descrito o mapeamento de Estelle sobre o o paradigma orientado a objeto.

3.1 Arquitetura da Ferramenta

A ferramenta para avaliar desempenho de sistemas distribuídos especificados em Estelle está representada na figura 3.2 onde são mostrados os módulos usados, suas entradas e saídas, desde a especificação Estelle até a cadeia de Markov gerada pela ferramenta (*RATES, STATES*). Nesta figura foram incluídos os módulos **Compilador** [6] e **Núcleo** [10] que são usados pela ferramenta. O módulo GEREST e as saídas EST, RATES e STATES serão descritos nas seções seguintes.

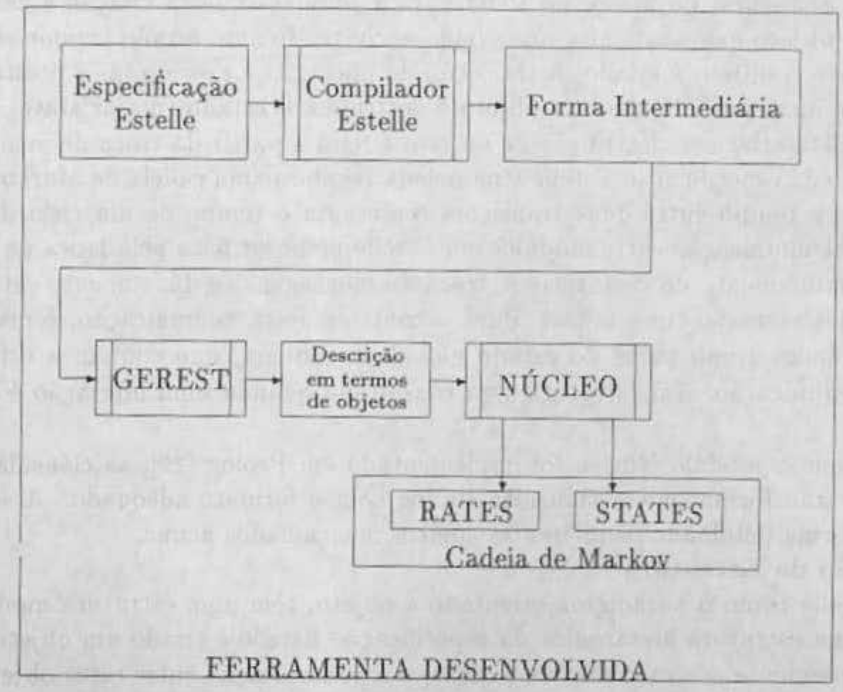

Figura 3.2: Estrutura da Ferramenta Desenvolvida

De forma geral, a ferramenta aceita como entrada uma especificação Estelle. Esta entrada é compilada (pelo Compilador Estelle) produzindo uma Forma intermediária que constitui a entrada do módulo GEREST. Este módulo é encarregado de gerar um arquivo que contém a especificação do sistema em termos de objetos e constitui a entrada do módulo Núcleo. A saída do Núcleo corresponde à cadeia de Markov que modela o sistema especificado.

3.2 De Estelle ao Modelo Orientado a Objeto

A ferramenta desenvolvida para avaliar desempenho de sistemas distribuídos especificados em Estelle mapeia as características de Estelle sobre as características do paradigma orientado a objeto (POO) descrito na seção 2. A partir desta descrição o módulo Núcleo gera a cadeia de Markov. São seis os tópicos mais importantes na conversão de uma especificação Estelle para o POO: Conversão da estrutura; Estado interno do módulo; Criação e destruição de módulos; Execução do sistema; Comunicação entre módulos; Conversão das cláusulas Estelle e Pascal.

A conversão da estrutura é feita de forma direta. Em outras palavras, existe uma relação um para um entre módulos Estelle e objetos do POO. Cada objeto, então, deve ter

seu comportamento interno modelado de acordo com a especificação Estelle do módulo que ele representa. Para que isto seja conseguido, o estado interno do objeto deve ser tal que represente o estado interno do módulo Estelle de forma completa, isto é, ele tem que incluir as variáveis definidas em Estelle, as filas *fifo* associadas aos pontos de interação, o estado (*major state*) em que se encontra o módulo, o nome do módulo pai, entre outros. É importante notar que ao incluir a informação do módulo pai no estado interno de um objeto, é obtida uma estrutura hierárquica com objetos, idêntica à formada com os módulos da especificação Estelle, tal como descrito na seção 2.

O POO [1] não inclui o instanciamento de objetos de forma dinâmica. Já que Estelle permite que a criação e destruição de módulos sejam feitas de forma dinâmica, foi implementada uma seqüência de passos no POO para a simulação desta criação e destruição de módulos. Um objeto que não tenha sido criado encontra-se num estado (*major state*) **INATIVO**. Quando o objeto é criado, a transição de iniciação é executada, e o estado (*major state*) inicial é alcançado. Quando o objeto é destruído, o estado (major state) volta a ser **INATIVO**. Esta criação e destruição de objetos é feita a partir da troca de mensagens.

A execução da especificação Estelle é modelada segundo uma cadeia de Markov de tempo discreto, onde o tempo entre duas transições representa o tempo de um ciclo de execução em Estelle. A comunicação entre módulos em Estelle pode ser feita pela troca de mensagens e pelo compartilhamento de variáveis. A troca de mensagens se dá, sómente entre módulos com pontos de interação conectados. Para o controle desta comunicação, é mantida uma estrutura de dados como parte do estado global do sistema, que contém a estrutura dos enlaces de comunicação. Tal estrutura será consultada quando uma interação é enviada ou recebida.

Uma vez que o módulo Núcleo foi implementado em Prolog [29], as cláusulas Estelle e Pascal foram transformadas em cláusulas Prolog com o formato adequado. A seguir serão descritos de forma detalhada os diferentes tópicos mencionados acima.

Conversão da Estrutura

Tanto Estelle como o paradigma orientado a objeto, têm uma estrutura modular. Para cada módulo na estrutura hierárquica da especificação Estelle é criado um objeto com prioridade correspondente aos módulos em Estelle, e a comunicação entre estes objetos se dará sómente entre os objetos que representam módulos com pontos de interação conectados. O controle da comunicação é descrito nos parágrafos seguintes. A descrição do comportamento de cada objeto é feita de acordo com a especificação Estelle. Esta especificação é feita usando a troca de mensagens e a definição de transições espontâneas no paradigma orientado a objeto. Por exemplo, na especificação do produtor–consumidor da seção 2, são descritos três objetos: o primeiro corresponde ao módulo especificação; o segundo ao módulo produtor; e o terceiro ao módulo consumidor.

Estado interno do Objeto

O estado interno de cada objeto é formado por sete componentes que contêm o estado interno do módulo Estelle representado. Estes componentes são, por exemplo, as variáveis definidas em Estelle, as filas *fifo* associadas aos pontos de interação, entre outros. O exemplo seguinte que corresponde ao módulo "Pro" do exemplo produtor–consumidor da seção 2, mostra a estrutura que representa o estado interno de cada objeto correspondente a um módulo Estelle:

```
S(IIPQ(NIL,NIL),CIPQ(NIL),EV(),PV(),LV(),INATIVO,*PAI).
```

onde: (1) IIPQ é o conjunto de filas *fifo* associadas aos pontos de interação definidos como filas individuais; (2) CIPQ é a fila *fifo* associada aos pontos de interação definidos como fila comum; (3) EV(...) é o conjunto de variáveis exportadas. Estas variáveis só são incluídas

no objeto que representa a raiz da especificação *"Root"* e a consulta e modificação destas variáveis é feita pelo envio de mensagens a este objeto; (4) PV(...) é o conjunto de variáveis de parâmetro; (5) LV(...) é o conjunto de variáveis locais; (6) INATIVO: representa o estado atual do objeto, que corresponde ao estado definido em Estelle (*major state*); (7) *PAI: nome do objeto pai. Esta variável permanece indefinida até que o módulo seja criado. Durante a criação esta variável será unificada com o nome do módulo pai.

Além destes componentes, o objeto raiz contém o conjunto de enlaces de comunicação que ligam os pontos de interação do sistema especificado.

Criação e Destruição de Módulos

Um módulo não criado encontra-se num estado *"indefinido"*. Quando criado (iniciado) pelo módulo pai, este módulo executa uma transição de iniciação, passando a ter um estado definido. Este estado inicial é especificado em Estelle pela cláusula *"initialize to"*. A situação em que se encontra um módulo antes de ser criado e após sua destruição é simulada no POO usando um novo estado que representa o estado indefinido em que se encontra este módulo. A criação de módulos é feita então pelo envio de uma mensagem especial *"INIT"* ao módulo que vai ser criado. O objeto sendo criado, ao receber esta mensagem, executa a sua transição de iniciação que é definida como uma transição de entrada no POO. Durante a execução da transição de iniciação, o módulo pai espera o módulo filho terminar a sua iniciação, podendo este (o módulo sendo criado), por sua vez, criar outros módulos enviando a mensagem *"INIT"* a seus módulos filhos. Quando a transição de iniciação é concluída, o primeiro ciclo de execução é iniciado.

A mensagem *"RELEASE"* é usada para destruir um módulo, levando-o ao estado *"INATIVO"*. Este estado representa o estado indefinido em que se encontra um módulo antes de ser criado e quando é destruído. Antes de atingir o estado inativo, o módulo desencadeia uma seqüência de destruição dos módulos filhos assim como dos seus enlaces de comunicação. Ou seja, toda a sub-árvore que tem como raiz o módulo destruído será também destruída.

A Execução do Sistema Especificado e o Modelo de Temporização Adotado para Estelle

A cada objeto é dada uma prioridade de acordo com a posição na árvore formada pela especificação, sendo o objeto raiz *"Root"* o de maior prioridade, de acordo com as prioridades de execução definidas por Estelle. Após o término de uma transição disparada no módulo raiz, um novo ciclo de execução é iniciado. No caso em que o módulo raiz não tem transições prontas para disparar, seus módulos filhos recebem permissão para executar uma transição. O mesmo processo é repetido caso eles não tenham transições sensibilizadas ou prontas para disparar. Já que a única forma de execução permitida neste trabalho é a não determinística, em cada ciclo de execução só estará sendo executada uma transição, podendo, durante o próximo ciclo ser executada qualquer transição sensibilizada de acordo com as normas de Estelle. Esta forma de execução pode ser modelada por uma cadeia de Markov de tempo discreto, usada para o cálculo das medidas de desempenho do sistema especificado.

Estelle dá liberdade na interpretação do conceito de tempo [15, 14, 2]. A única condição, é que o tempo tem que ser progressivo e uniformemente atualizado entre as transições temporizadas. Para manter as propriedades básicas dos processos Markovianos este tempo é considerado com distribuição geométrica. O tempo especificado na cláusula *"delay"* é considerado como o tempo médio de disparo. Uma transição sem a cláusula *"delay"* é considerada com tempo de permanência de um intervalo, ou em outras palavras, com probabilidade um o próximo estado a ser visitado será diferente do estado presente.

Este modelo de Estelle equivalente uma cadeia de Markov de tempo discreto pode ser estendido a uma cadeia de Markov de tempo contínuo. Como uma cadeia de Markov de tempo contínuo pode ser transformada em uma de tempo discreto pelo método de *"unifor-*

mização" [24], a representação por tempo discreto pode também representar um modelo de tempo contínuo.

Uma limitação que Estelle apresenta para a avaliação de desempenho é o fato de não permitir a especificação de probabilidades de disparo nas transições. Estelle [14] define que, quando duas transições estão prontas para disparar, uma delas será escolhida de forma arbitrária. Para eliminar este problema, uma função especial, a função *RANDOM(...)*, que nos permite especificar com que probabilidade uma transição é disparada, foi implementada em Estelle. Esta função associa uma probabilidade de disparo a uma transição. Caso esta função não seja especificada, as transições sensibilizadas são disparadas com uma distribuição de probabilidade uniforme. A função RANDOM(...) é usada junto a cláusula Estelle *"Provided"*, como mostrado nos exemplos da seção 5.

Comunicação entre Módulos

A comunicação entre dois módulos Estelle pode ser feita pela troca de mensagens e o compartilhamento de variáveis entre módulo pai e filho na estrutura hierárquica da especificação. Estes dois tipos de comunicação são simulados no paradigma orientado a objeto da forma descrita abaixo.

A troca de mensagens é feita mantendo uma estrutura de dados que contém o estado dos enlaces dos pontos de interação como parte do estado global do sistema. Esta estrutura é modificada pelas cláusulas Estelle *"attach, detach, connect, disconnect"*. Esta mesma estrutura é consultada quando uma cláusula *"output"* é executada, para se saber o ponto de interação e o objeto destino da interação enviada. Uma vez feita esta consulta, uma mensagem é enviada ao objeto destino, contendo como dados a interação e o nome do ponto de interação destino. O objeto destino, ao receber esta mensagem, a deposita na fila associada ao ponto de interação indicado na mensagem.

Para as variáveis compartilhadas também é mantida uma estrutura de dados contendo as variáveis compartilhadas dos módulos (*EV(...)*). Esta estrutura é modificada ou consultada mantendo a semântica de Estelle enviando mensagens ao objeto *"raiz"*.

Estas estruturas de dados são mantidas somente no objeto *"raiz"* da especificação, para simplificar a estrutura do estado interno do objeto, já que, caso contrário, cada objeto teria que armazenar a estrutura dos enlaces de comunicação dos módulos filhos, assim como o estado das variáveis compartilhadas, complicando o seu manejo.

Para cada um dos casos citados anteriormente (Troca de mensagens, compartilhamento de variáveis criação e destruição de módulos), define-se, no paradigma orientado a objeto, um evento de entrada. As demais transições descritas em Estelle, são especificados como transições ou eventos espontâneos no paradigma orientado a objeto, podendo disparar sempre que as suas condições sejam satisfeitas. Este disparo é feito com uma dada probabilidade.

Conversão das cláusulas Estelle e Pascal para Prolog

Uma vez que a ferramenta de análise de desempenho de protocolos especificados em Estelle foi desenvolvida em Prolog, deve-se transformar as cláusulas Estelle e Pascal em cláusulas Prolog. Os aspectos mais importantes desta conversão são apresentados nos parágrafos seguintes. É importante notar, porém, que esta metodologia é independente da linguagem de implementação.

A conversão das cláusulas Estelle *"output, init, release, connect, disconnect, attach, detach"* é feita de forma direta, implementadas com o envio de mensagens pelos objetos que geram a transição como indicado acima. Por exemplo a cláusula Estelle *"output p1.INT"* é representada pela cláusula Prolog: envie_mensagem(int(Cons,p1,INT),Root).
Esta cláusula representa a forma de especificar o envio de mensagens no paradigma orientado a objeto e indica o envio da mensagem *"int(Cons,p1,INT)"* ao objeto "Root". No exemplo mostrado acima, a mensagem é formada pelo nome do componente (Cons), o nome do ponto

de interação (p1) e a interação que se está enviando (INT). O destino inicial da mensagem é o módulo raiz (Root), já que a estrutura de dados que contém a informação do estado dos enlaces de comunicação encontra-se neste módulo. Este objeto, ao receber a mensagem, consulta a estrutura de dados e determina que módulo e que ponto de interação são os verdadeiros destinatários da informação, enviando-lhes, a seguir, a mensagem. Para o objeto destinatário da informação é definida uma ação para a recepção desta mensagem. A informação é armazenada na fila *fifo* associada ao ponto de interação correspondente. Esta ação é definida da seguinte forma:

MENSAGEM int(Cons,P2,int):(IIPQ(FP1,FP2),...) → (IIPQ(FP1,FP2_N),...);
AÇÃO :append(FP2,int,FP2_N);

Nesta ação especifica-se a reação do objeto destinatário ao receber a mensagem como uma interação. Esta reação, em forma geral, corresponde à determinação da fila *fifo* associada ao ponto de interação referenciado na mensagem, ao armazenamento da interação nesta fila (append(FP2,int,FP2_N)) e à determinação do novo estado atingido quando esta ação é executada. Para implementar as outras cláusulas Estelle são usados procedimentos similares ao descrito acima para a cláusula *"output"*. Os trechos em Pascal da especificação são implementados de acordo com a semântica da linguagem Pascal.

3.3 O Módulo de Geração (GEREST)

O módulo GEREST (figura 3.2) é o encarregado de fazer a conversão da forma intermediária de Estelle para a especificação orientada a objeto. A saída deste módulo é um arquivo contendo um conjunto de cláusulas Prolog que constituem a estrutura de uma dada especificação Estelle em termos de objetos e formam a entrada do módulo Núcleo. Por exemplo, fazem parte deste arquivo o estado inicial do sistema que, é representado da forma mostrada no seguinte exemplo:

```
OBJETO : Root
ESTADO : S(IIPQ(),CIPQ(),EV(E(Cons,EV()).E(Pro,EV()).NIL),PV(),LV(),
         CHCIPF(NIL),INATIVO,VAZIO)

OBJETO : Cons
ESTADO : S(IIPQ(NIL,NIL),CIPQ(NIL),EV(),PV(),LV(FLAG(*)),INATIVO,*PAI)

OBJETO : Pro
ESTADO : S(IIPQ(NIL,NIL),CIPQ(NIL),EV(),PV(),LV(),INATIVO,*PAI)
```

Neste exemplo a especificação é formada por três módulos, o módulo "Root", o módulo "Cons" e o módulo "Pro". No módulo Root, a estrutura de dados (CHCIPF(...)) é iniciada com uma lista vazia, já que nenhum ponto de interação foi conectado ainda. No módulo "Cons" a lista de variáveis locais LV(...) não é vazia e contém uma variável FLAG(*). Da mesma forma, a estrutura de dados que contém as filas *fifo* não é vazia e contém duas filas inicialmente vazias (NIL). Um outro ponto a ser destacado é o estado (*major state*) definido em Estelle que é iniciado como INATIVO.

Além da informação descrita acima, o arquivo contém, por exemplo, informações sobre as instâncias de objetos criados, entre outras. Estas informações indicam a relação existente entre cada instância e o corpo que contém a forma de execução. Esta informação tem o seguinte formato: TIPO(Cons,Conbody).

Existem transições que são fixas para toda especificação. Por exemplo, a forma de receber uma interação, o recebimento da mensagem de iniciação e de destruição, bem como a

descrição das cláusulas Estelle connect, disconnect, attach entre outras. Estas cláusulas podem ser descritas de forma comum para todos os módulos e usar as caracteristicas do Prolog (backtrack) para a sua execução.

4 Exploração do Espaço de Estados

Um modelo de um sistema complexo pode dar origem a um grande número de estados (milhões ou mesmo bilhões de estados) da cadeia de Markov correspondente, o que torna difícil (ou mesmo impossível) a análise exata de tais sistemas. Uma análise aproximada do sistema especificado torna-se indispensável nestes casos. É claro que a aproximação deve ser tal que o resultado final esteja dentro de margens de erro aceitáveis.

Existem várias técnicas na literatura que abordam o problema da explosão do espaço de estados em modelos Markovianos. Por exemplo, Muntz *et al* [21], obtém limites superiores e inferiores para uma função recompensa em estado estacionário, utilizando técnicas de agregação de estados a partir de um certo número de estados explorados. A utilidade deste método foi demostrada para modelos de disponibilidade.

Em [8], foi proposto um método de exploração dinâmica de estados. Em outras palavras, o objetivo daquele trabalho é o de gerar apenas os estados de maior probabilidade (em estado estacionário) da cadeia de Markov do sistema. O método baseia-se no cálculo recursivo do número médio de visitas a cada estado da cadeia.

Neste trabalho utilizamos também uma técnica de exploração dinâmica para obter os estados mais prováveis. O trabalho é baseado no algorítmo de [8] Entretanto, diferente de [8], desenvolvemos um algorítmo iterativo para o cálculo dos parâmetros que guiam a geração do próximo estado da cadeia. O objetivo final deste novo algorítmo é gerar um conjunto de estados tal que o tempo médio de permanência neste conjunto seja maior do que um dado valor. Os estados são escolhidos de forma a atingir o objetivo com o menor número de estados possível. Maiores detalhes podem ser encontrados em [7], incluindo exemplos que mostram os ganhos obtidos com esse novo algorítmo em relação ao algorítmo de [8].

5 Exemplos

Nesta seção são apresentados dois exemplos simples que demostram o utilização de uma especificação em Estelle para avaliação de desempenho.

Produtor–Consumidor

Este exemplo foi usado nas seções anteriores e aqui só será apresentado o resultado da analise da especificação. A análise do exemplo com a ferramenta descrita neste artigo obtém como resultado cinco estados com as respectivas probabilidades de transição. Esta saída tem a seguinte forma (os dados X(0.0,0.0,0.0098,0.98,0.0098) representam as probabilidades estacionárias de cada estado) :

estado inicial	estado final	probabilidade de disparo
0	1	1.0
1	2	1.0
2	3	1.0
3	4	0.01
3	3	0.99
4	2	1.0

X(0.0,0.0,0.0098,0.98,0.0098)

Protocolo Stop and Wait

O protocolo especificado aqui é uma versão do protocolo stop and wait [27]. A estrutura da especificação é mostrada na figura 5.3, e abaixo fornecemos uma especificação informal deste protocolo.

A especificação deste protocolo é formada por três módulos: Relogio; Transmissor; Receptor. O módulo "Relogio" é o encarregado de controlar o tempo de retransmissão de mensagens. Quando uma mensagem vai ser enviada, o módulo "Transmissor" inicia o "Relogio" com a mensagem **"start_timer"**. 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 "Relogio" e entrando em um estado de processamento de informação, para em seguida transmitir a mensagem com um dado número de seqüência. O módulo "Receptor", ao receber uma mensagem, a processa e envia uma mensagem de reconhecimento (ack) ao módulo "transmissor". Esta mensagem pode ser corrompida na rede. O módulo de rede especifica uma rede de transmissão que pode perder mensagens. A especificação Estelle deste protocolo encontra-se em [18].

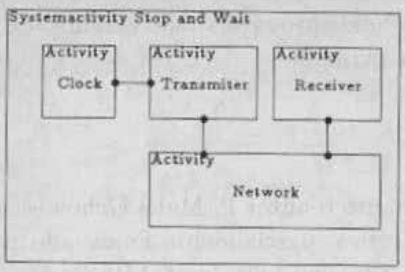

Figura 5.3: Estrutura da especificação Stop and Wait

Esta especificação gera um número finito de estados, uma vez que um "controle" evita o prenchimento irrestrito das filas *fifo* associadas aos pontos de interação. Em outras palavras, estas filas possuem, em princípio, "buffer" infinito, mas a especificação deste protocolo é feita utilizando-se de "buffer" finito.

Quando esta especificação foi analisada, foram encontrados 61 estados. Quando o algorítmo de exploração dinâmica de estados descrito na seção anterior é utilizado, encontra-se 45 estados, para um tempo mínimo de permanência especificado em 1.0E7. Neste caso, é possível verificar que os estados com maior probabilidade estacionária são aqueles onde o transmissor e o receptor estão processando as mensagens trocadas. Embora a redução de estados não seja significativa neste exemplo, é possível se obter reduções significativas do número de estados gerados, em modelos bem mais complexos.

A lista com as probabilidades obtidas para cada estado é apresentada a seguir: X0 = 0; X1 = 0.0010849768735; X2 = 0.0021656225019; X3 = 2.1656225019E-6; X4 = 0.49218693226; X5 = 0.0019705192477; X6 = 1.9648180928E-4; X7 = 1.9701252226E-6; X8 = 0.0019692366096; X9 = 3.9353373405E-7; X10 = 0.44820545002; X11 = 3.9349376111E-7; X12 = 0.0017924633074; X13 = 1.7931703969E-4; X14 = 0.0017921048864; X15 = 3.5856337043E-7; X16 = 0.0017917465371; X17 = 3.5849217386E-7; X18 = 8.9587326857E-4; X19 = 3.5842074064E-7; X20 = 9.8240904642E-5; X21 = 0.045025918413; X22 = 9.0231118511E-5; X23 = 9.0410364598E-5; X24 = 1.082811251E-9; X25 = 1.0817284397E-6; X26 = 1.0795693011E-6; X27 = 4.4775573242E-4; X28 = 3.9377938929E-10; X29 = 1.7906647967E-6; X30 = 8.9551146483E-8; X31 = 1.7903067354E-6; X32 = 1.7906647967E-10; X33 = 1.7899487456E-6; X34 = 1.7903067354E-10; X35 = 8.9497437281E-10; X36 = 1.7878012366E-6; X37 = 1.7899487456E-10; X38 = 1.7878012366E-10; X39 = 1.0795693011E-9; X40 = 1.9676686703E-7; X41 = 3.9331698072E-7; X42 = 1.7921037032E-7; X43 = 1.7921037032E-7; X44 = 1.9676686703E-10; X45 = 3.5842074064E-7;
MTTF = 10647935.408; CPU TIME = 93159.

Outros exemplos de análise de protocolos mais complexos podem ser encontrados em [18].

6 Conclusões

A linguagem de especificação formal de sistemas distribuídos Estelle (FDT definida pela ISO), tem sido muito utilizada para a especificação de protocolos de comunicação. Assim sendo, muitas ferramentas que oferecem suporte a tais desenvolvimentos tem sido apresentadas na literatura. Particularmente, a avaliação de desempenho de protocolos especificados em Estelle é um tópico pouco explorado, tendo-se conhecimento de apenas um trabalho para tal fim, baseado na simulação de Estelle com parâmetros de tempo [22].

A ferramenta descrita neste trabalho é uma contribuição para a análise de desempenho de protocolos de comunicação de dados e sistemas distribuídos em geral. A partir de uma especificação em Estelle, a cadeia de Markov associada é gerada automaticamente e em forma simbólica. A utilização de um novo algorítmo de exploração de estados permite que apenas os estados "mais prováveis" sejam gerados a partir de um estado inicial. O cálculo das medidas de interesse pode ser feito então, de forma exata (todos os estados são gerados) ou aproximada (apenas os estados "mais prováveis" são gerados, de acordo com uma tolerância especificada), a partir da cadeia gerada.

Agradecimentos

Este trabalho foi realizado enquanto o autor P. Mejía Ochoa estava na Universidade Federal do Rio de Janeiro, COPPE/Elétrica, parcialmente financiado pela Empresa Hondureña de Telecomunicaciones Hondutel e por uma bolsa da CAPES.

Referências

[1] S. Berson, E. de Souza e Silva, and R.R. Muntz. An object oriented methodology for the specification of Markov models. In *The First International Conference on the Numerical Solution of Markov Chains*, pages 2–29, 1990.

[2] S. Budkowski and P. Dembinski. An introduction to estelle: A specification language for distributed systems. *Computer Networks and ISDN Systems*, Março 1987.

[3] J.P. Courtiat. How could estelle become a better fdt. In H.Rudin and C.H. West, editors, *Protocol Especification Testing and Verification*, 1987.

[4] J.P. Courtiat. Estelle* - a powerful dialect of estelle for osi protocol description. Technical report, SEDOS, 1988.

[5] J.P. Courtiat, P. Dembinski, R. Groz, and C. Jard. Estelle : Un langage iso pour les algorithmes distribués et les protocoles. Technical report, INRIA, 1986.

[6] R.C. de Oliveira Jr. Um compilardor para a linguagem estelle. Master's thesis, Universidade Federal do Rio de Janeiro, 1991.

[7] E. de Souza e Silva and P. Mejía. State space exploration in markov models. *relatório técnico UFRJ/NCE*, 1991.

[8] D.D. Dimitrijevic and M.S. Chen. An integrated algoritm for probabilistic protocol verification and evaluation. Technical report, IBM, 1988.

[9] M.C. Diniz and E. de Souza e Silva. Especificação e geração de modelos markovianos para análise de desempenho e confiabilidade de sistemas. *Submetido a publicação*, 1991.

[10] M.C. Diniz and E. de Souza e Silva. Uma ferramenta para especificação e geração de modelos markovianos. In 8º *Simpósio Brasileiro de Redes de Computadores*, pages 15–36, 1991.

[11] J.R. Engelbrecht, P.S. Kritzinger, and H. Rudin. Predicting protocol performance from a meta–implementation. In M.Diaz, editor, *Protocol Especification Testing and Verification*, 1986.

[12] M.A. Holliday and M.K. Vernon. A generalized timed petri net model for performance analysis. *IEEE Transactions on Software Engineering*, Dezembro 1987.

[13] ISO. *ISO/SC21/WG1/FDT–A Guidelines For the Application of Estelle, LOTOS and SDL Project/97.21.9/Q48.2 CCITT/SG/X/Q7*,, 1987.

[14] ISO. *ISO/TC97/SC21/WG1/DIS9074 ESTELLE - A Formal Description Technique Based on an Extended State Transition Model*, 1987.

[15] R.J. Linn. jr. The features and facilities of estelle. In *Protocol Specification Testing and Verification, VII*, 1987.

[16] T.W. Page Jr., S.E. Berson, W.C. Cheng, and R.R. Muntz. An object-oriented modeling environment. *ACM Sigplan Notices (Proceedings OOPSLA'89)*, 24(10):287–296, 1989.

[17] P.S. Kritzinger. Protocol performance using image protocols. In H.Rudin and C.H. West, editors, *Protocol Especification Testing and Verification*, 1987.

[18] P.E. Mejía. Avaliação de desempenho de sistemas distribuídos especificados em estelle. Master's thesis, Universidade Federal do Rio de Janeiro, 1990.

[19] M. Menasche and B. Berthomieu. Time petri nets for analyzing and verifying time dependent comunication protocols. In H.Rudin and C.H. West, editors, *Protocol Especification Testing and Verification*, 1983.

[20] M.K. Molloy. Performance analysis using stochastic petri nets. *IEEE Transactions on Computers*, Setembro 1982.

[21] R.R. Muntz, E. de Souza e Silva, and A. Goyal. Bounding availability of repairable computer systems. *IEEE Transactions on Computers*, Dezembro 1989.

[22] P.Dembinski and S.Budkowski. Simulating estelle specifications with time parameters. In H.Rudin and C.H. West, editors, *Protocol Especification Testing and Verification*, 1987.

[23] A.C.P. Pedroza and P.R.O. Valim. Um simulador de protocolos de comunicação para a linguagem estelle. In *VII Simpósio Brasileiro de Telecomunicações*, 1988.

[24] S.M. Ross. *Stochastic Processes*. John Wiley and Sons, California, 1983.

[25] H. Rudin. From formal protocol specification towards automated performance prediction. In H.Rudin and C.H. West, editors, *Protocol Especification Testing and Verification*, 1983.

[26] H. Rudin. An improved algoritm for estimating protocol performance. In Y. Yemini, R. Strom, and S. Yemini, editors, *Protocol Especification Testing and Verification*, 1985.

[27] M. Schwartz. *Telecommunication Networks: Protocols, Modeling and Analysis.* Addisson-Wesley Publishing Company, Massachusetts, 1987.

[28] R. Sijclmassi and P. Gaudette. An object oriented model for estelle. In K.J. Turner, editor, *Formal Description Techniques*, 1989.

[29] L. Sterling and E. Shapiro. *The Art of Prolog.* Mit-Press, Massachusetts, 1986.

[30] B. Walter. Timed petri–nets for modeling and analyzing protocols with real–time´characteristics. In H.Rudin C.H. West, editor, *Protocol Especification Testing and Verification*, 1983.