UM SISTEMA PARA GERAÇÃO AUTOMÁTICA DE SEQUÊNCIAS DE TESTES PARA PROTOCOLOS DE COMUNICAÇÃO

Gutembergue Soares da Silva [1,3,4]

Aloysio de Castro Pinto Pedroza [1,2]

[1] **COPPE/UFRJ**
Programa de Engenharia Elétrica
Centro de Tecnologia, Bloco H
Cidade Universitária
Caixa Postal 68504
21945 Rio de Janeiro RJ

[2] **EE/UFRJ**
Departamento de Eletrônica

[3] **UFRN**
Departamento Eng. Elétrica
Centro de Tecnologia

[4] **TELERN**
Departamento de Engenharia

RESUMO

Este trabalho apresenta um sistema de geração automática de sequências de teste para protocolos de comunicação, baseado em uma técnica para testar a conformidade da componente de controle de protocolos e serviços, modelada como sistemas de transição. A idéia chave da técnica é o conceito de uma marca única, denominada Sequência Única de Entrada/Saída, derivada para cada estado da especificação do protocolo. Estas sequências são usadas para gerar sequências de teste que percorrem cada transição de estado. São abordados com maior ênfase o procedimento de geração de teste, a arquitetura do sistema e aspectos relativos a definição de níveis de conformidade, com suas implicações na geração de sequências de testes para protocolos reais. Como exemplos, são apresentados modelos de máquinas de estados finitas de uma entidade de protocolo simples e do protocolo do Bit Alternado, juntamente com as sequências de testes obtidas com auxílio do sistema.

I. INTRODUÇÃO

Os protocolos de comunicação, em particular aqueles padronizadas por organismos internacionais, tais como a ISO e o CCITT, são, em geral, bastante complexos e demandam um esforço considerável para sua implementação. A especificação de um protocolo padrão é um documento detalhado que descreve interfaces e mecanismos do protocolo. Implementações incorretas de aspectos de um protocolo podem resultar em implementações incompatíveis, ainda que derivadas do mesmo padrão. Assim, faz-se necessário testar cada implementação de

um dado protocolo para verificar sua conformidade com uma especificação padrão. Uma parte integrante do teste de conformidade é a geração de sequências de teste a partir da especificação do protocolo considerado.

Em função da complexidade dos protocolos, um teste exaustivo que assegure a derivação de todas as possíveis sequências de interação é pràticamente inviável. De outra forma, testes aleatórios com o propósito de diminuir o número de sequências através de escolhas arbitrárias podem sofrer, no que se refere aos aspectos de completude e eficiência, justificadas restrições. Diante destas considerações, alinhadas em [1], a elaboração de sequências de testes eficazes assume um papel relevante nas atividades de teste de protocolos.

Este trabalho apresenta um sistema de geração automática de sequências de testes para protocolos de comunicação, baseado em uma técnica para testar a conformidade da componente de controle de protocolos e serviços, modelada como sistemas de transição.

A seção 2 apresenta quatro técnicas para geração de sequências de teste, os métodos T, U, D e W, e discute, brevemente, aspectos relativos ao comprimento das sequências produzidas e à cobertura de falhas.

Na seção 3 é descrito o método utilizado para geração das sequências de teste, com ênfase na modelagem da especificação do protocolo a ser testado, no procedimento de geração das sequências de teste e na caracterização da sequência única de entrada/saída.

A seção 4 descreve o sistema desenvolvido, realizando uma abordagem de sua arquitetura e principais características dos módulos componentes.

Na seção 5, como exemplo de utilização do sistema, são mostradas as sequências de teste resultantes da sua aplicação em modelos de máquinas de estados finitas.

Finalizando, na seção 6, estão apresentadas algumas observações conclusivas.

II. TÉCNICAS DE GERAÇÃO DE SEQUÊNCIAS DE TESTE

As técnicas de derivação de sequências de testes podem ser agrupadas em duas classes [2]: uma classe caracterizada pela **varredura de transições**, que inclue todas as transições da Máquina de Estados Finita (MEF), pelo menos uma vez, e uma outra referente aos métodos que exigem que a MEF possua uma **entidade de caracterização**, ou seja, uma sequência especial de interações, como meio de identificar os estados da máquina.

Recentemente, Sidhu e Leung realizaram um estudo, apresentado em [3], sobre aplicações de quatro métodos formais (os métodos T, U, D e W) para geração de sequências de testes de protocolos, com destaque para a estimativa de cobertura de falhas.

O **método T (Método de Varredura de Transições)** é relativamente simples, comparado aos outros três métodos apresentados a seguir. Este método produz uma sequência de entradas que são aplicadas na Máquina de Estados Finita em seu estado inicial e percorre cada transição da MEF, pelo menos uma vez. O Método admite que a MEF é fortemente conectada; detecta os erros na função de saída mas não detecta os erros na função de próximo estado [4].

A sequência de teste gerada pelo **Método U (Método da Sequência Unica)** apresentado em [5] percorre todas as transições de estado e deriva uma marca única para cada estado, denominada Sequência Unica de Entrada/Saída (sequência UIO). A sequência UIO para um estado corresponde a um comportamento que não é exibido por nenhum outro estado. A capacidade de detecção de falhas das sequências de testes geradas por este procedimento depende do nível de conformidade do teste.

O **Método D (Método da Sequência de Distinção)** tem como idéia chave encontrar uma Sequência de Distinção [6], isto é, um conjunto de entradas que geram diferentes saídas para cada estado da MEF. A sequência de verificação é obtida pela concatenação de uma sequência que conduz a MEF ao estado inicial, uma sequência que reconhece os estados e uma sequência que reconhece todas as transições. As sequências geradas por este método asseguram a detecção de erros na função de saída e na função de próximo estado.

Para aplicação do **Método W (Método da Sequência de Caracterização)** é necessário a construção de dois conjuntos: o Conjunto W, que possui sequências de entradas tais que a saída observada pela aplicação de W é diferente para cada estado, e o Conjunto P, que contém todos os percursos parciais da árvore de testes construída a partir do grafo da MEF. A concatenação dos conjuntos P e W origina uma sequência de teste chamada de Sequência de Caracterização [7]. Este método assegura a detecção de erros na função de saída e na função de próximo estado.

No contexto de definição de teste de protocolos, o conceito de **cobertura de falhas** refere-se à potencialidade de uma sequência de teste, para verificar se uma implementação conforma sua especificação. Para avaliar a abrangência de falhas ou erros é necessário determinar o conjunto de máquinas de estados finitas, que não são equivalentes à especificação mas que, quando submetidas à sequência de teste, produzem as mesmas saídas. Desde que a não equivalência destas máquinas, em relação à especificação, não é identificada pela sequência de teste, a abrangência de detecção de falhas se torna restrita, pois nem todas as falhas podem ser detectadas [8].

As comparações entre os métodos de geração de sequências, incluem, principalmente, aspectos relacionados com a resposta a diferentes tipos de erros e ao comprimento das sequências produzidas. Podem ser destacadas as seguintes comparações, estabelecidas em [3], entre os quatro métodos :

- as capacidades de detecção de falhas das sequências produzidas pelos métodos U, D e W são equivalentes e superiores à do método T;
- o método T é o que produz a sequência de teste de menor comprimento;
- o comprimento de uma sequência de teste do método U é menor que uma do método D e este, por sua vez, menor que uma sequência produzida pelo método W.

Um aspecto relevante sobre a aplicabilidade dos diferentes métodos diz respeito ao requisito de **complementação da máquina de estados finita da especificação**. Para os métodos D e W esta é uma condição necessária e suficiente para geração das entidades de caracterização, respectivamente, a sequência de distinção e o conjunto W. Enquanto que, para o método U, a condição é somente suficiente (não necessária) para geração de sua entidade de caracterização (sequências UIO).

Um ponto comum aos métodos abordados é que não há garantia de unicidade para as sequências de testes por êles produzidas. Para o método T, isto é evidente pela natureza aleatória de sua geração. Para os métodos U , D e W, isto decorre do fato de que pode existir mais do que uma entidade de caracterização para a mesma MEF.

III. A METODOLOGIA PARA GERAR AS SEQUÊNCIAS DE TESTES

Os métodos T, U, D e W podem ser usados com maior eficácia, dependendo do enfoque de verificação do teste e da estratégia de sua utilização.

Este trabalho, nesta fase em particular , utiliza basicamente o **método U** e em determinados casos um conceito semelhante àquele da entidade de caracterização do **método W**.

III.1 O MODELO

Os **sistemas de transição simples** também conhecidos como Máquinas de Estado Finitas tem sido amplamente utilizados para modelar a parte de controle de protocolos e serviços [2].

Uma Máquina de Estados Finita é representada como um grafo direcionado **G=(V,A)** onde **V** é o número vértices e **A** o número de arestas. Os vértices representam os estados da MEF e cada aresta, associada a uma designação **i/o**, caracteriza uma operação de entrada e uma operação de saida, que corresponde a uma possível transição de estado. Os elementos **i** e **o** pertencem, respectivamente, ao **conjunto I** de operações de entradas e ao **conjunto O** de operações de saída.

São requisitos necessários à máquina de estados para aplicação do método U: a definição de um **estado inicial fixo**, a **existência de um percurso** deste estado a qualquer estado da especificação, a apresentação em sua **forma mínima**, o atributo de ser **fortemente conectada** e sua caracterização como uma **máquina do tipo Mealy**, o que garante uma saída para cada transição, em função do estado da máquina e da entrada.

A apresentação em sua **forma mínima** proporciona um padrão efetivo e permite que uma implementação correta de uma máquina seja utilizada como referência para testar outras. O atendimento ao requisito de **conectividade forte** assegura que a máquina tem todos os estados alcançáveis.

A máquina pode ser levada a seu estado inicial, a partir de qualquer estado, pela aplicação de uma **entrada de "reset"** que gera, para esta transição, uma saída nula. As transições com entradas de "reset" náo fazem parte da máquina original; entretanto, a introdução de tais transições garantem o atendimento ao requisito da máquina ser fortemente conectada.

A filosofia do método de teste utilizado encara a implementação como uma caixa-preta, com acesso limitado a uma porta de entrada e uma porta de saída. Os componentes de uma sequência são fornecidos como entradas, à caixa-preta, por uma estrutura denominada **testador**, que observa as saídas e compara-as com um padrão esperado. Se a implementação produz as saídas esperadas, é comprovada sua conformidade com a especificação. O procedimento de teste adotado abstrai de detalhes a arquitetura de teste. Não serão considerados na geração das sequências de testes os problemas de sincronismo relatados em [9], uma vez que nessa primeira abordagem o trabalho direciona-se ao teste local da especificação do protocolo, com vistas à simulação.

Os exemplos 1 e 2 , apresentados a seguir, serão utilizados para ilustrar alguns conceitos da metodologia e a aplicação do sistema

EXEMPLO 1.

A especificação de uma entidade de protocolo modelado como uma máquina de estados finita é ilustrada pelo grafo apresentado na fig. 1. Na tabela de transições da máquina (tabela 1) estão contidas as informações relativas aos estados,

entradas, saídas e interações, definidas para este protocolo. O estado inicial é **0** e a entrada de "reset" é **r**. Neste exemplo a máquina possui 6 estados, cinco entradas e 4 saídas. Cada estado possui uma transição para o estado inicial, com a designação **(r/N)**, não representada na a fig. 1. A letra **N** simboliza a saída nula.

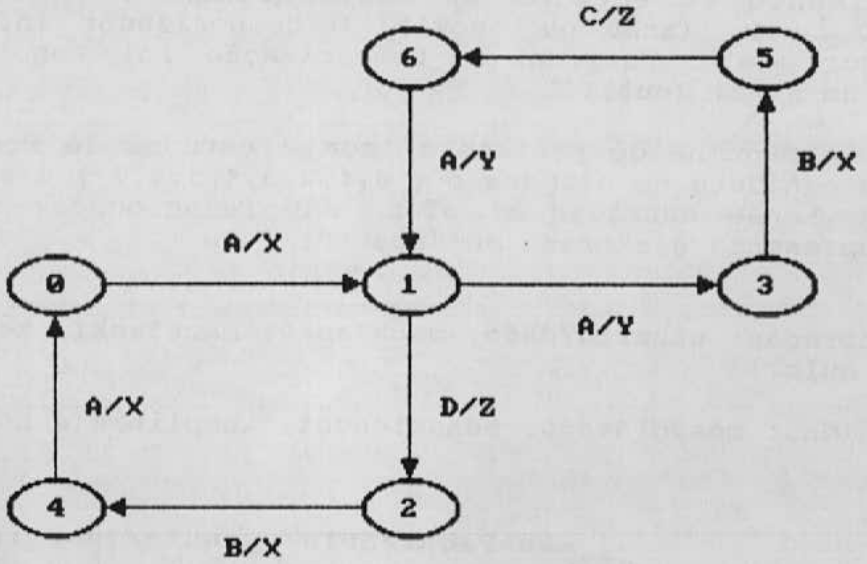

Fig.1 Diagrama de transição de estado para a MEF do Exemplo 1

	saídas				próximo estado			
entrada / estado	A	B	C	D	A	B	C	D
0	X	-	-	-	1	-	-	-
1	Y	-	-	Z	3	-	-	2
2	-	X	-	-	-	4	-	-
3	-	X	-	-	-	5	-	-
4	X	-	-	-	0	-	-	-
5	-	-	Z	-	-	-	6	-
6	Y	-	-	-	1	-	-	-

Tabela 1 : Tabela de Transições para a máquina da fig.1

EXEMPLO 2.

O protocolo do **Bit Alternado (ABP)** é frequentemente utilizado como exemplo de um protocolo porque, apesar de sua relativa simplicidade, possui muintas características usadas em protocolos reais. Este exemplo corresponde a seção de transmissão do ABP. O transmissor interage com um processo

usuário **(usuário)**, um meio de menssagens **(mdado)**, um processo temporizador **(temp)** e um meio de reconhecimento **(mack)**. Para manter o sequenciamento das menssagens e de seus respectivos reconhecimentos, são utilizados os valores **0** e **1**. O transmissor recebe menssagens **(dado)** do processo usuário e as envia numeradas com com valores **0** ou **1** **(dado0** ou **dado1)**. Do meio de reconhecimento é recebido um reconhecimento numerado com valores **0** ou **1** **(ack0** ou **ack1)**. O temporizador informa ao transmissor que o período de temporização foi esgotado pelo envio de um sinal **(out)**.

A máquina do protocolo, com 8 estados, é mostrada na fig. 2. O conjunto de estados é **{ 0,1,2,3,4,5,6,7 }** e seu estado inicial é **0**. As entradas e saídas são relacionadas a seguir, onde **r** representa a entrada de "reset":

- Entradas: **usuário?dado, mack?ack0, mack?ack1, temp?out, r e nulo.**
- Saídas: **mdado!dado0, mdado!dado1, temp!inic e nulo.**

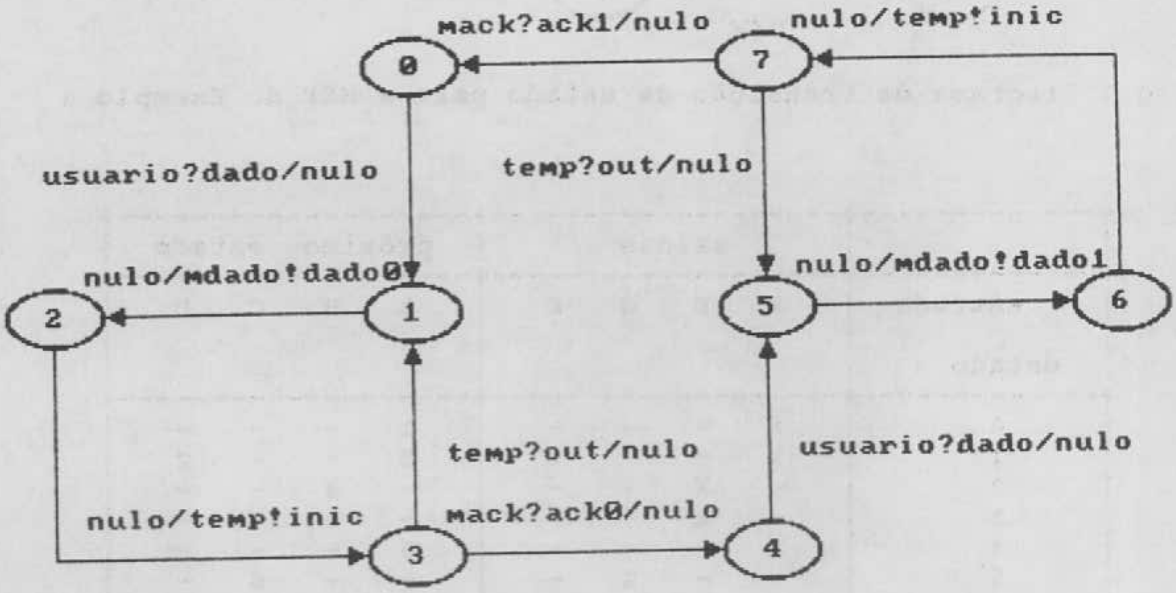

Fig.2: Diagrama de transição de estado para a MEF do Exemplo 2

Para combinações de entrada_estado que não estão definidas no **comportamento essencial** do protocolo, ou seja, na especificação que corresponde à sua funcionalidade básica, a entidade do protocolo produzirá uma saída nula e permanecerá no estado atual. Deste modo, deve existir, para cada estado, uma aresta correspondente a uma entrada que é ignorada. Isto caracteriza a chamada **suposição de completude**. Considerando que a maioria dos protocolos não é especificada completamente, a conformidade é definida em dois níveis [11]:

- **Conformidade Forte:** uma implementação tem conformidade forte em relação à sua especificação se ambas geram as mesmas saídas para todas as sequências de entrada;

- **Conformidade Fraca:** uma implementação tem conformidade fraca em relação à sua especificação, se a implementação tem o mesmo comportamento entrada/saída que a especificação do protocolo composta únicamente de arestas essenciais, mas possui um comportamento não especificado para combinações estado_entrada originadas das arestas que constituem o comportamento não essencial do protocolo.

Uma outra alternativa para caracterizar a **suposição de completude** é a criação de um estado de erro para a MEF de modo que sempre que uma entrada não especificada no comportamento essencial for recebida, uma saída de erro é gerada, indicando que uma entrada inesperada para aquele estado foi recebida. Uma vez no estado de erro, a especificção deverá ignorar todas as entradas até receber uma entrada de "reset".

Em seguida, serão apresentadas algumas definições necessárias para discussões posteriores. Para cada aresta **a** do grafo que representa a MEF, são definidas três funções :

- **Origem (a):** refere-se ao estado inicial da aresta;

- **Final(a):** denota o estado final da aresta;

- **Designação (a):** corresponde ao evento entrada/saída da transição associada.

Uma **sequência** é definida como uma lista finita e ordenada de pares entrada/saída. As variáveis **n** e **m** representam o número de estados e o número de transições da MEF, respectivamente.

III.2 AS SEQUÊNCIAS ÚNICAS DE ENTRADA/SAÍDA (UIO)

O **método U** caracteriza-se, essencialmente, pela derivação de uma marca única de entrada/saída para cada estado da especificação. Uma **sequência UIO** para um estado é um comportamento entrada/saída que não é exibido por qualquer outro estado. Além de distinguí-lo de todos os outros estados, a **sequência UIO** verifica se a implementação evolui corretamente para o próximo estado, após a execução da transição.

Uma das principais vantagens das sequências UIO é que na prática, normalmente, as Máquinas de Estados Finitas possuem sequências UIO para cada um dos seus estados. As sequências de entrada destas entidades de caracterização podem ser diferentes para os vários estados da MEF, de modo que estes estados podem

ser identificados tanto pelas entradas, como pelas saídas que elas geram. Assim sendo, se as sequências de entrada das sequências UIO para todos os estados forem idênticas, as sequências de saídas produzidas deverão ser diferentes [10].

III.3 O PROCEDIMENTO DE GERAÇÃO DO TESTE

O procedimento de obtenção da sequência UIO para cada estado reside basicamente em se verificar a unicidade de sequências de entrada/saída que tem seu comprimento progressivamente aumentado, num processo cíclico de propagação, até a sequência UIO ser encontrada.

Inicialmente, são construídas e verificadas a unicidade de sequências de entrada/saída de comprimento unitário. Em caso de insucesso, o procedimento é repetido na construção de todas as sequências de comprimento maior que um, partindo do estado considerado, até a sequência UIO ser encontrada ou o comprimento das sequências atingir, progressivamente, o limite de $2n^2$

Este procedimento, descrito em [11], baseia-se na construção de dois conjuntos , constituídos de três vetores cada, para armazenamento de todas as informações inerentes às fases do processo de busca da sequência UIO.

No primeiro vetor da fase inicial, são armazenadas todas as sequências distintas de comprimento unitário geradas para um estado. Como componentes do segundo vetor, são armazenados, em correspondência, os estados finais associados a cada sequência construída. No terceiro vetor, são armazenados os estados que, além do estado de trabalho, possuem a sequência de teste correspondente alocada no primeiro vetor. Os componentes deste vetor são conjuntos de tuplas, onde o primeiro elemento de cada tupla é o estado de origem e o segundo, o estado final a que a máquina é levada, após sofrer a aplicação da referida sequência.

Na segunda fase, são construídas sequências de comprimento 1 a partir das sequências de comprimento (1-1) e as informações similares são armazenadas no segundo conjunto de vetores, cujos elementos são construídos com a mesma estrutura dos anteriores. Após a realização de um ciclo de busca da unicidade da sequência, caso não ocorra sucesso, tais informações são transferidas para o primeiro conjunto de vetores e reinicializado o ciclo, com comprimento das sequências incrementado de uma unidade.

A localização da sequência única, para os diversos estados da máquina, é obtida pela aplicação de um algoritmo no elenco de vetores associados a cada estado de trabalho. No caso de sequências de comprimento unitário, é examinada a condição de inexistência de elementos, nos conjuntos formados por arestas da do grafo associado à MEF, rotuladas com a mesma

designação das arestas saintes do estado sob análise.

Para sequências de comprimento maior que a unidade, é examinada a mesma condição, nos conjuntos formados por arestas construídas a partir de arestas armazenadas no correspondente conjunto do ciclo predecessor. Tais arestas possuem o mesmo estado inicial das arestas do referido conjunto, e o estado final coincidente com o destino das arestas originadas nos seus respectivos estados finais.

Como exemplo da aplicação do Método, são ilustrados em seguida alguns passos para o cálculo da sequência UIO referente ao estado 6 da máquina do Exemplo 1. O conjunto de arestas saintes do estado 6 é {(6,1)}. A Designação (A/Y) desta aresta não é única, uma vez que o conjunto O1={ {(6,1),(1,3)}-{(6,1)}}, composto de arestas da MEF que também possuem esta Designação, não é vazio. O ciclo de busca da unicidade para sequências de comprimento maior que a unidade é iniciadao pela construção do conjunto {(1,2),(1,3)}, formado pelas arestas saintes do estado final da aresta de trabalho. Para cada aresta que constitui este conjunto, extraí-se sua Designação e constroí-se uma sequência pela concatenação com (A/Y), que resulta nas sequências (A/Y),(D/Z) e (A/Y),(A/Y).

Com relação a estas sequências é examinada a condição de desigualdade entre seus elementos finais e a Designação da aresta (3,5), formada a partir do estado final da aresta (1,3), elemento do conjunto O1. A condição mencionada é verdadeira para primeira sequência, pois (D/Z) é diferente de (B/X), e a busca da sequência UIO pode ser interrompida. Deste modo, (A/Y),(D/Z) partindo do estado 6, o identifica unicamente, pois esta sequência específica não é exibida por nenhum outro estado. De outro modo, se a entrada é a sequência (A/D) e a saída a sequência (Y/Z), sabe-se que a máquina encontrava-se no estado 6 antes da aplicação da sequência de entrada.

A complexidade do algoritmo envolvido no procedimento descrito acima é, no pior caso, O ($n^2(tmax)^{(2n^2+2)}$), onde tmax é o número máximo de transições de saída de qualquer estado. Se um estado não possui uma sequência que não excede ao comprimento limite $2n^2$ calculada do modo explicitado acima, uma fórmula alternativa é oferecida para construção da sequência UIO, onde a distinção dos estados remanescentes é feita entre cada par de estados [11].

O procedimento para gerar a sequência de teste consiste na concatenação de subsequências de teste para cada transição. A sequência realiza uma inspeção através de uma varredura de todas as transições, com propósito de verificar a existência de cada estado e transição na implementação, além da designação correta de cada transição.

Para cada aresta do grafo, associada a uma transição, a sequência de teste leva a máquina ao estado inicial, encontra o menor percurso deste estado ao estado inicial da aresta, aplica uma entrada que provoca a transição de estado e finaliza aplicando a sequência UIO ao estado final da aresta.

IV. DESCRIÇÃO DO SISTEMA PARA GERAÇÃO AUTOMATICA DE SEQUÊNCIAS DE TESTE PARA PROTOCOLOS DE COMUNICAÇÃO

O objetivo deste sistema é fornecer ao projetista de protocolos uma ferramenta de concepção assistida por computador que, com base na descrição formal de protocolos, possa realizar de forma automatizada a tarefa de gerar sequências de testes.

As diversas funções do sistema estão encapsuladas em diferentes módulos componentes que lhe dão suporte, quais sejam: o módulo de **edição da especificação**, o de **geração de sequências UIO** e o módulo de **geração de sequências de testes**. Com apoio comum de um **módulo básico**, estes módulos atuam sob coordenação de um módulo **gerente**. A estrutura funcional dos módulos pode ser vista como uma estrutura em camadas, onde cada um utiliza, como entrada, produtos elaborados pelos módulos de execução precedente.

IV.1 ARQUITETURA DO SISTEMA

A arquitetura do Sistema é mostrada na fig. 3, onde são destacados os seguintes componentes :

- **Editor Dedicado** : este módulo proporciona ao usuário o máximo de flexibilidade e conforto, durante a fase de edição da MEF que representa o protocolo, produzindo uma especificação de forma estruturada, que será fornecida como entrada para os demais módulos;

- **Gerador de Sequências UIO**: módulo com atribuição de gerar sequências de entrada/saída para identificar de forma única cada estado da especificação;

- **Gerador de Sequências de Teste**: com o objetivo de gerar e otimizar sequências de teste, este módulo realiza suas tarefas a partir de entradas produzidas pelos dois módulos anteriores, através de cinco submódulos :

  . **Gerador de Transições de "Reset"**: submódulo que agrega transições de "reset" à máquina para garantir seu retorno ao estado inicial, após testar um estado ou transição, habilitando-a para a execução de outro teste;

  . **Submódulo de Complemento da Especificação**: vinculado à opção de teste de conformidade forte, é ativado opcionalmente para inserir transições relativas às entradas não especificadas para os estados da máquina objetivando complementar sua especificação;

  . **Gerador de Menor Caminho** : encontra o menor caminho da origem para todos os estados da MEF;

. **Gerador de Varredura de Transições** : constrói subsequências de varredura para cada transição da MEF com objetivo de testar a sua correção na aplicação das funções origem, designação e final;

. **Submódulo de Otimização de Sequências de Teste:** faz comparações para verificar, na sequência de teste, a existência de subsequências completamente contidas em outras e as elimina.

- **Gerente** : responsável pelo acesso aos módulos executivos de edição da especificação, geração de sequências UIO e geração de sequências de teste;

- **Módulo Básico** : contém funções de acesso a estruturas de dados e funções básicas de formatação de tela e impressão, bem como funções de carga e salvamento do contexto de todo trabalho realizado.

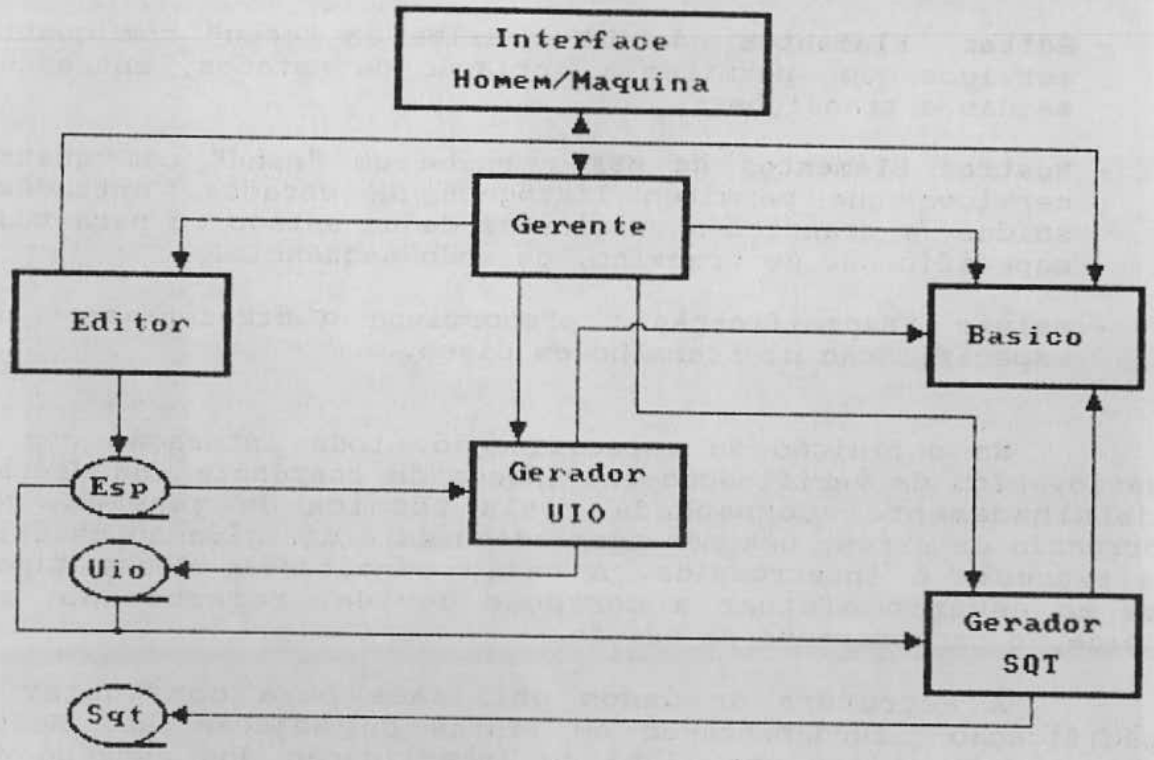

Fig. 3: Arquitetura Funcional do Sistema

IV.2 EDIÇÃO DA MÁQUINA DE ESTADOS

O módulo de **edição da especificação** permite, a partir de um diálogo estabelecido com o usuário, uma definição ágil do protocolo como uma máquina de estados finita, através de um sistema orientado por "menus" com sintaxe de fácil compreensão. Esta interface realiza, durante a edição, diversas verificações visando manter a coerência dos dados e eliminar erros de construção fornecendo avisos de alerta ao usuário.

A edição é ativada a partir da escolha de uma das opções disponíveis em "menus". Estes "menus" estão estruturados hierárquicamente, de modo que, após selecionar uma determinada opção, um novo conjunto de opções poderá ser exibido na tela. A apresentação de "menus" ao usuário é feita através de mecanismos de janela.

Visando proporcionar um acompanhamento confortável da edição, estão disponíveis comandos para definição, de forma modular, dos parâmetros e elementos da especifiacção, tais como:

- **Editar Elementos da MEF** : exibe um "menu" com quatro serviços que permitem a criação de estados, entradas, saídas e transições;

- **Mostrar Elementos da MEF** : exibe um "menu" com quatro serviços que permitem listagens de estados, entradas, saídas e transições originadas de um estado ou para toda especificação de trabalho, de modo sequencial;

- **Salvar Especificação** : proporciona o armazenamento da especificação de trabalho em disco.

Na definição da especificação, toda interação com o usuário, além da verificação intrínseca de coerência dos dados, é detalhadamente acompanhada pela técnica de janelas. Na ocorrência de erros, uma mensagem adequada é dirigida ao usuário e a execução é interrompida. A cada sinalização deste tipo, cabe ao usuário efetuar a correção devida, reiniciando, em seguida, o procedimento de edição.

A estrutura de dados utilizada para configurar a especificação fundamenta-se em listas de adjacências. Nesta representação existe uma lista de interligação dos estados da máquina, cujos itens, por sua vez, são listas que armazenam as transições para os estados que lhe são adjacentes. Cada relação possui, portanto, um ítem de cabeça. Os ítens de cabeça são armazenados sequencialmente, permitindo um fácil acesso à lista de adjacências de um determinado estado.

Em alguns casos os ítens foram condensados em alocação sequencial nas listas de adjacências, por meio de uma transformação simples, eliminando-se os campos de ligação. Esta forma de alocação é útil quando os estados da máquina devem

sofrer um processamento na mesma ordem que estão dispostos na alocação sequencial. É o caso com o qual nos deparamos quando é necessário realizar o armazenamento permanente da especificação.

Assim sendo, para manipulações futuras da estrutura a representação com os campos de ligação é retomada. A matriz de adjacências é uma outra estrutura que pode ser utilizada para armazenar o grafo da máquina de estados. Esta representação, entretanto, introduz um certo desperdício de memória para grafos esparsos e demanda maior tempo dos algoritmos de manipulação do que as listas de adjacências, onde somente arestas existentes no grafo são representadas.

IV.3 GERAÇÃO DAS SEQUÊNCIAS UIO

O módulo de **geração de sequências únicas** tem como função principal derivar da especificação uma marca única para cada estado, construindo sua sequência única de entrada/saída.

Neste módulo são usadas listas encadeadas de ligação simples com o propósito atender os requisitos de flexibilidade para acomodação de sequências de comprimento variável 1, a priori desconhecido, bem como as informações correlatas.

Os conjuntos definidos para dar suporte à geração das sequências UIO são implementados como listas encapsuladas, que admitem um elenco restrito de operações, tais como : armazenamento, busca, exibição e deleção. A complexidade do ciclo de detecção da unicidade da sequência e a natureza dinâmica destes conjuntos, cujos elementos são permanentemente atualizados durante a geração da sequência, condicionou a utilização da alocação dinâmica no processo, com o objetivo de racionalizar o uso da memória.

Este módulo apresenta opções de evolução passo a passo ou automática. O usuário terá acesso visual, interativamente, às informações correntes (estado de trabalho, transição em curso, caminho percorrido, etc.). As tabelas 2 e 3 apresentam o resultado correspondente à atuação deste módulo nas máquinas dos Exemplos 1 e 2.

estado	sequência UIO
0	(A/X,D/Z)
1	(D/Z)
2	(B/X,A/X)
3	(B/X,C/Z)
4	(A/X,A/X)
5	(C/Z)
6	(A/Y,D/Z)

Tabela 2 : Sequências UIO para a máquina da fig.1

estado	sequência UIO
0	(usuário?dado/nulo,nulo/mdado!dado0)
1	(nulo/mdado!dado0)
2	(nulo/temp!inic,temp?out/nulo,nulo/mdado!dado0)
3	(temp?out/nulo,nulo/mdado!dado1)
4	(usuário?dado/nulo,nulo/mdado!dado1)
5	(nulo/mdado!dado1)
6	(nulo/temp!inic,temp?out/nulo,nulo/mdado!dado1)
7	(temp!out/nulo,nulo/mdado,dado1)

Tabela 3 : Sequências UIO para a máquina da fig.2

IV.4 GERAÇÃO E OTIMIZAÇÃO DAS SEQUÊNCIAS DE TESTES

O módulo de **geração e otimização de sequências de testes** realiza basicamente as seguintes tarefas:

- deriva, a partir do grafo da MEF, as sequências de "reset", que levam a especificação a seu estado inicial;
- calcula o menor percurso do estado inicial a todos os estados da MEF;
- constrói as subsequências de varredura, que percorrem todas as transições de cada estado da especificação, incluindo as de "reset";
- verifica a existência de subsequências de teste completamente contidas em outras.

Duas alternativas, relacionadas com o nível de conformidade do teste desejado, são oferecidas ao usuário : **teste de conformidade fraca** e **teste de conformidade forte**. Na primeira, são executadas exatamente as tarefas relacionadas acima. Na segunda, além destas, é ativado um procedimento com o objetivo de complementar a máquina de estados finita, que modela a especificação. Este procedimento de completude foi implementado para atuar sobre a especificação original, de modo que para cada entrada não especificada em um estado, uma transição é criada para esta entrada. Estas transições artificiais produzem saídas nulas e a máquina permanece no estado atual.

Outra maneira de implementar a complementação é introduzir um **estado de erro**. Sempre que uma entrada não especificada é recebida, uma saída nula é gerada indicando que uma entrada inesperada para aquêle estado foi recebida. O estado de erro deve ignorar todas as entradas até receber uma entrada de "reset".

O procedimento de cálculo do menor percurso da origem a todos os estados da MEF recorre ao algoritmo "Shortest Path" aplicado a grafos dirigidos apresentada em [12]. A implementação do processo de busca dos menores caminhos é suportada por listas encadeadas.

As **subsequências de varredura** têm a finalidade de verificar a correção de cada transição, no que se refere à sua origem, designação e final. Sua construção é precedida de constatações sobre a existência dos conjuntos que fornecem elementos para sua composição.

O nível de conformidade do teste define o número de transições envolvidas no procedimento de varredura. No teste de conformidade fraca, somente as transições essenciais do protocolo são percorridas. Por ourto lado, no teste de conformidade forte, que pressupõe a máquina completamente especificada, todas as transições são envolvidas, o que corresponde a varrer um número de transições igual ao produto da quantidade de estados da máquina **n**, pela quantidade de **entradas** definidas no **conjunto I** de entradas da MEF.

A **sequência de teste**, na forma não otimizada, é construída pela concatenação de uma sequência de "reset", com as subsequências geradas para cada transição da MEF, através do procedimento de varredura. Deste modo, esta sequência é composta por **m** subsequências, onde **m** é o número de transições.

A **otimização da sequência de teste** é obtida pela análise e eliminação de subsequências que estão completamente contidas em outras.

V. APLICAÇÕES E RESULTADOS

Estão relatados a seguir os resultados da aplicação do sistema na geração de sequências de teste para os exemplos 1 e 2, nos dois níveis de conformidade.

Para o Exemplo 1, o módulo **gerador de sequências UIO** produz as sequências únicas de entrada/saída para cada estado apresentadas na tab. 2. Considerando que a máquina do exemplo não é completamente especificada, foi gerada, inicialmente, uma sequência de teste para verificar a conformidade fraca. A sequência de teste produzida pelo módulo gerador de sequências de teste, na forma não otimizada, consiste de 15 subsequências listada na fig. 5. Submetidas à atuação do procedimento de otimização, esta sequência sofre eliminação de 6 subsequências que encontram-se inteiramente contidas em outras, constituindo-se na sequência de teste otimizada, listada na fig. 6, com às 9 subsequências remanescentes.

[r/N,A/X,D/Z],[r/N,A/X,D/Z,B/X,A/X],
[r/N,A/X,A/Y,B/X,C/D],
[r/N,A/X,D/Z,B/X,A/X,A/X],
[r/N,A/X,D/Z,B/X,A/X,A/X,D/Z],
[r/N,A/X,A/Y,B/X,C/D],
[r/N,A/X,A/Y,B/X,C/D,A/Y,D/Z],
[r/N,A/X,A/Y,B/X,C/D,A/Y,D/Z],
[r/N,r/N,A/X,D/Z],
[r/N,A/X,r/N,A/X,D/Z],
[r/N,A/X,D/Z,r/N,A/X,D/Z],
[r/N,A/X,A/Y,r/N,A/X,D/Z],
[r/N,A/X,D/Z,B/X,r/N,A/X,D/Z],
[r/N,A/X,A/Y,B/X,r/N,A/X,D/Z],
[r/N,A/X,A/Y,B/X,C/D,r/N,A/X,D/Z]

Fig. 5: Sequências não otimizadas para testar a conformidade fraca da máquina do Exemplo 1.

[r/N,A/X,D/Z,B/X,A/X,A/X,D/Z],
[r/N,A/X,A/Y,B/X,C/D,A/Y,D/Z],
[r/N,r/N,A/X,D/Z],
[r/N,A/X,r/N,A/X,D/Z],
[r/N,A/X,D/Z,r/N,A/X,D/Z],
[r/N,A/X,A/Y,r/N,A/X,D/Z],
[r/N,A/X,D/Z,B/X,r/N,A/X,D/Z],
[r/N,A/X,A/Y,B/X,r/N,A/X,D/Z],
[r/N,A/X,A/Y,B/X,C/D,r/N,A/X,D/Z]

Fig. 6: Sequências otimizadas para testar a conformidade fraca da máquina do Exemplo 1.

Como forma de atender a suposição de completude na realização do teste de conformidade forte, na máquina do Exemplo 1, é necessário adicionar "self-loops" aos estados com entradas não especificadas, o que corresponde a complementar a especificação para obter o comportamento não essencial do protocolo. Como decorrência, o número de transições envolvidas no procedimento de varredura sofre um aumento expressivo de 15 para 35. Nestas condições, a sequência de teste otimizada, derivada de um elenco de 35 subsequências, resulta em 29, das quais 20 subsequências, entre elas, testam se a implementação ignora as entradas não especificadas, conforme estabelecido na suposição de completude.

Desde que os protocolos reais raramente são completos, a complementação de uma máquina do protocolo, como mencionado acima, em geral, introduz um grande número de transições artificiais, resultando, portanto, em sequências de teste extremamente longas. Para o exemplo em questão, o teste de conformidade forte produziu um número de subsequências três vêzes maior que o teste de conformidade fraca.

A aplicação do módulo de **geração de sequências UIO** na máquina do Exemplo 2, que modela a seção de transmissão do protocolo de do Bit Alternado, resultou nas sequências UIO mostradas na tabela 3. Considerando que a saída nula não fornece uma indicação precisa dos estados da máquina, foi assegurado, na construção das sequências UIO, a existência de pelo menos um elemento com saída não nula. Com o objetivo de testar a máquina deste modelo a nível de conformidade fraca, foi gerada uma sequência que, na forma não otimizada, contém 18 subsequências. A fig. 7 apresenta a sequência de teste otimizada, resultante da eliminação de 7 subsequências. O teste de conformidade forte também pode ser executado para o este modelo, desde que o requisito de completude seja satisfeito, através do complemento da especificação.

```
[r/nulo,usuário!dado/nulo,nulo/mdado!dado0,nulo/temp!inic,
 temp?out/nulo,nulo/mdado!dado0],
[r/nulo,usuário!dado/nulo,nulo/mdado!dado0,nulo/temp!inic,
 mack?ack0/nulo,usuário!dado/nulo,nulo/mdado!dado1,
 nulo/temp!inic,temp?out/nulo,nulo/mdado!dado1],
[r/nulo,usuário!dado/nulo,
 nulo/mdado!dado0,nulo/temp!inic,mdado?ack0/nulo,
 usuário!dado/nulo,nulo/mdado!dado1,nulo/temp!inic,
 mack?ack1/nulo,usuário!dado/nulo,nulo/mdado!dado0],
[r/nulo,r/nulo,usuário?dado/nulo,nulo,mdado!dado0],
[r/nulo,usuário?dado/nulo,r/nulo,
 usuário?dado/nulo,nulo/mdado!dado0],
[r/nulo,usuário?dado/nulo,nulo/mdado!dado0,
 r/nulo,usuário?dado/nulo,nulo/mdado!dado0],
[r/nulo,usuário?dado/nulo,nulo/mdado!dado0,nulo/temp!inic,
 r/nulo,usuário?dado/nulo,nulo/mdado!dado0],
[r/nulo,usuário?dado/nulo,nulo/mdado!dado0,nulo/temp!inic,
 mack?ack0/nulo,r/nulo,usuário?dado/nulo,nulo/mdado!dado0],
[r/nulo,usuário?dado/nulo,nulo/mdado!dado0,
 nulo/temp!inic,mack?ack0/nulo,usuário?dado/nulo,
 r/nulo,usuário?dado/nulo,nulo/mdado!dado0],
[r/nulo,usuário?dado/nulo,nulo/mdado!dado0,
 nulo/temp!inic,mack?ack0/nulo,usuário?dado/nulo,
 r/nulo,usuário?dado/nulo,nulo/mdado!dado0],
[r/nulo,usuário?dado/nulo,nulo/mdado!dado0,nulo/temp!inic,
 mack?ack0/nulo,usuário?dado/nulo,nulo/mdado!dado1,
 nulo/temp!inic,r/nulo,usuário?dado/nulo,nulo/mdado!dado0]
```

Fig. 7: Sequências otimizadas para testar a conformidade fraca da máquina do Exemplo 2.

VI. CONCLUSÕES

Neste trabalho foi apresentada uma metodologia para gerar sequências de teste e um sistema, nela baseado, que permite ao usuário gerar, de forma automática, sequências de teste, a partir de uma especificação modelada como uma máquina

de estados finita. Estas especificações são introduzidas com facilidade e de maneira interativa, de modo que protocolos especificados em diferentes linguagens possam ser testados, dentro da metodologia adotada.

A versão atual do sistema, para microcomputadores compatíveis com IBM-PC/XT/AT, encontra-se implementada em linguagem PASCAL. A filosofia modular adotada para seu desenvolvimento configurou-lhe uma arquitetura onde a estrutura funcional dos módulos foi hierarquizada em camadas, de maneira que cada módulo utiliza elementos produzidos pelos anteriores.

A atuação do sistema nos exemplos utilizados para ilustrar a metodologia de teste permitiu caracterizar o problema do teste de conformidade forte em protocolos reais. As sequências de teste resultantes são extremamente longas em decorrência da introdução de trasições artificiais, para atender ao requisito de completude na sua realização.

Além dos exemplos apresentados, o sistema foi aplicado com resultados satisfatórios, demonstrando sua viabilidade em protocolos simples (Abracadabra, Transporte Simplificado etc.), estimulando, assim, estudos sobre soluções alternativas nos procedimentos de geração de sequências, visando melhorar a rapidez de resposta dos módulos de geração de sequências UIO e de geração de sequências de teste, para sua utilização em protocolos de maior complexidade.

Outro aprimoramento em sua concepção, de propósito imediato, diz respeito à sua integração a um Sistema de Auxílio ao Projeto de Protocolos [13], em fase final de desenvolvimento. A partir da Forma Intermediária [14], traduzida da especificação em ESTELLE de um dado protocolo, serão geradas sequências de teste para utilização durante as fases de simulação e teste final da sua implementação.

A médio prazo, uma proposta mais abrangente refere-se à integração das técnicas de teste fundamentadas em entidades de caracterização, que identificam os estados da MEF, em um único procedimento básico de geração e otimização de sequências de teste. Os trabalhos de Sidhu e Leung [3] são uma referência nessa direção.

REFERÊNCIAS

[1] H. Ural,"A Derivation Method for Protocol Conformance Testing",IFIP TC6, Protocol Specification, Testing and Verification : VII, 1987.

[2] B. Sarikaya,"Conformance Testing : Archtectures and Test Sequences", Computer Networks and ISDN Systems, Vol 17, 1989.

[3] Deepinder P. Sidhu e Ting-Kau Leung," Formal Methods for Protocol Testing: A Detailed Study", IEEE Trans. on Software Eng. Vol 15. N.4, 1989.

[4] S. Naito e M. Tsunoyama,"Fault Detection for Sequencials Machines by Transitions Tours", Proc. 11th IEEE Fault Tolerant Comput Conf. 1981.

[5] k. Sabnani e A. Dahbura, " A Procedure for Generating Protocol Tests", Proc. 9th Data Communications Syposium, 1985.

[6] Z. K ohavi, Switching and Finite Automata Theory, McGraw Hill, 1978

[7] T Chow,"Testing Software Design Modeled by Finite-State Machine", IEEE Trans. on Software Eng. Vol.4, 1978

[8] Deepinder P. Sidhu e Ting-Kau Leung,"Experience with Test Generation for Real Protocols", Proc. SIGCOMM '88 Symp. Commmunication Architetures and Protocols, 1988.

[9] B. Sarikaya e G.Bochmann,"Synchronization and Specification Issues in Protocol Testing", IEEE Trans. on Communications, Vol 32,1984.

[10] W.Y.L Chan, S.T. Vuong e M.R. Ito, " An improved Protocol Generation Procedure Based on UIOS", Proc. SIGCOMM Symp. Commmunication Architetures and Protocols, 1989

[11] K. Sabnani e A. Dahbura, " A Protocol Test Generating Procedure", Computer Networks and ISDN Systems, Vol 15, 1988.

[12] M. Syslo, N. Deo e J. Kowalik, Discrete Optimization Algorithms, Prentice-Hall, 1983.

[13] A. Pedroza, P.Valim, C. Goulart e R. Oliveira, "Um Sistema de Auxílio ao Projeto de Protocolos de Comunicação para Redes de Computadores", Seminário Franco Brasileiro em Sistemas Informáticos Distribuídos,1989.

[14] J.Courtiat e J. Ayache,"Design Specification of ESTELLE Intermediate Form", Rapport SEDDOS 410/ LAAS/81/6,May/86.