UTILIZAÇÃO DA TÉCNICA ESTELLE PARA GERAR SEQÜÊNCIAS DE TESTE PARA PROTOCOLOS DE COMUNICAÇÃO

Vitório Bruno Mazzola, Luís Otávio Rodrigues da Silva

Departamento de Engenharia Elétrica
Universidade Federal de Santa Catarina
88040-900 - Campus Universitário - Florianópolis - SC
Tel.: (048) 231-9506 - Fax.: (048) 231-9770 - E-mail: mazzola@eel.ufsc.br

RESUMO. Este artigo apresenta uma proposta de utilização de ESTELLE e de ferramentas de suporte para a geração de seqüências para o Teste de Conformidade a partir de especificações formais de protocolos de comunicação. Uma característica interessante do método proposto é a automatização quase completa do processo de geração de seqüências de teste, reduzindo sua complexidade. A aplicação do método sobre um exemplo simples permite ilustrar os benefícios obtidos.
PALAVRAS CHAVE. Protocolo de Comunicação, Teste de Conformidade, Técnicas de Descrição Formal, Seqüências de Teste.

ABSTRACT. This paper introduces a methodology based on the ESTELLE FDT for generating Conformance Testing Suites from formal specifications of communication protocols. An interesting feature of the proposed approach is the automatization of several steps in the generation process, reducing the complexity. Application on a simple example permits verify the obtained benefits.
KEYWORDS. Communication Protocols, Conformance Testing, Formal Description Techniques, Test Cases.

1. INTRODUÇÃO

A definição do modelo de referência OSI, [1] e a normalização de um extenso conjunto de protocolos de comunicação foram, sem dúvida, grandes passos dados em direção ao desenvolvimento das redes de computadores.

Apesar do esforço que vem sendo desenvolvido desde então, uma série de problemas restam a resolver e podemos destacar aqueles relacionados ao desenvolvimento dos protocolos de comunicação compondo uma dada arquitetura. A grande complexidade de tais protocolos tem motivado o desenvolvimento de trabalhos voltados à definição de modelos e metodologias para o seu desenvolvimento. Podemos destacar aí a aplicação de modelos baseados em sistemas de transição de estados, como as máquinas de estado finitos (*finite state machines, FSM)*, [2], as Redes de Petri, [3], e a definição de técnicas de descrição formal (*formal description techniques, FDT*), feitas no seio dos organismos internacionais de normalização. Exemplos destas técnicas são ESTELLE e LOTOS definidas pela ISO, [4], [5] e SDL definida pelo CCITT, [6].

Estas técnicas têm desempenhado, nos últimos anos, um papel bastante importante no que diz respeito à concepção de protocolos de comunicação e mesmo de outras classes de aplicações distribuídas. Ferramentas de software, desenvolvidas em torno destas técnicas, permitem hoje automatizar as principais etapas de concepção de um protocolo ou de uma aplicação distribuída.

Uma etapa de grande importância no ciclo de vida de um protocolo de comunicação é o Teste de Conformidade. É através da realização desta etapa que uma implementação vai ser confrontada no que diz respeito à sua coerência com a norma que definiu o protocolo em questão.

A realização do teste de conformidade é baseada na aplicação, à implementação, de um conjunto de dados de entrada, ou simplesmente *entradas* e a verificação, com base no

conjunto de dados obtidos na saída, ou *saídas*, se a implementação apresenta ou não o comportamento esperado.

A etapa de teste de conformidade, pela sua importância, deve ser caracterizada por dois parâmetros, de certo modo, antagônicos: a *eficiência*, que sugere um menor dispêndio de tempo possível na realização desta etapa; e a *eficácia*, que está relacionada ao grau de cobertura ou abrangência da atividade de teste.

Assim sendo, o teste de conformidade da implementação de um protocolo deve ser *eficiente* e *eficaz*. É evidente que, dado o antagonismo destas duas definições, a satisfação de uma delas pode resultar no desrespeito total à outra definição. A solução, neste caso, é encontrar o maior equilíbrio possível entre estes dois parâmetros, o que, evidentemente, não é uma tarefa das mais simples.

Uma coisa, no entanto é clara. A obtenção de um equilíbrio satisfatório entre *eficiência* e *eficácia* está fortemente ligada à metodologia empregada para a obtenção do conjunto de dados a ser aplicado à implementação. A esta atividade é dado o nome de *geração de seqüências de teste* e, para isto, o uso de métodos baseados em técnicas formais pode ser a chave para a obtenção de um conjunto de seqüências que determinem o perfeito equilíbrio entre eficiência e eficácia.

O trabalho enfocado neste artigo permitiu definir uma metodologia de geração de seqüências de teste de conformidade de protocolos de comunicação, metodologia esta baseada na utilização dos mecanismos presentes na técnica ESTELLE e nas suas ferramentas de suporte.

A seção 2 apresenta os conceitos introdutórios no que diz respeito ao teste de protocolos de comunicação; a seção 3 descreve os principais métodos encontrados na bibliografia para a derivação de seqüências de teste a partir de especificações formais; nesta seção, as técnicas formais que servem de base para a geração de teste são também apresentadas; na seção 4, é apresentada a metodologia de geração de seqüências de teste proposta neste trabalho, assim como as ferramentas desenvolvidas para suportar tal metodologia; um exemplo de aplicação da metodologia proposta é apresentada na seção 5; finalmente, a seção 6 apresenta as conclusões relativas a esta experiência.

2. OS TESTES DE CONFORMIDADE: PRINCIPAIS CONCEITOS

A especificação de um protocolo é baseada no serviço de comunicação a ser proporcionado e o serviço de comunicação que o protocolo irá utilizar. Os projetistas devem checar a especificação do protocolo para garantir que ele é correto e consistente, proporciona o serviço de comunicação desejado e oferece serviços com uma eficiência aceitável. O protocolo deve satisfazer às regras de sua especificação tão bem quanto qualquer restrição no seu ambiente. A implementação, que pode ser em hardware, software ou uma combinação de hardware e software, consiste na geração de uma representação concreta e executável vinda de uma descrição formal e abstrata.

Quando o comportamento de uma implementação está de acordo com as regras da especificação, então a implementação está **conforme** à especificação. A maneira mais viável de determinar a conformidade é testar a implementação com um conjunto de testes.

Os testes de conformidade são importantes porque asseguram que implementações do mesmo protocolo, geradas independentemente, podem trabalhar em conjunto. Os testes de conformidade iniciam com o teste do projeto, no qual é desenvolvido um pacote de testes abstrato (ATS: Abstract Test Suite). Um pacote de testes é uma coleção de casos de testes, cada um dos quais é utilizado para testar um aspecto do protocolo.

O desenvolvimento de um pacote de teste pode ser manual ou semi-automático. O projeto manual é a maneira convencional para se projetar um pacote de teste, o projetista deve ter

experiência no padrão do protocolo e ser familiar com a estrutura e metodologia dos testes de conformidade, porém, o projeto manual é complicado e susceptível a erros.

Atualmente, é largamente aceito que os testes de conformidade OSI são cruciais para a conclusão dos objetivos definidos pela norma OSI. Pacotes de teste padrão necessitam ser desenvolvidos para cada protocolo padrão, sendo que a padronização dos pacotes de teste de conformidade necessita ser baseada numa metodologia de testes padrão e num modelo para especificação do pacote de testes.

A realidade mostra que os testes de conformidade são uma questão técnica difícil, acompanhada de alguns problemas. Diferentes soluções para estes problemas levam a maneiras diferentes para se conduzir os testes de conformidade. Contudo, todos os métodos de teste atentam para acessar aspectos estáticos e dinâmicos da implementação de um protocolo com respeito a critérios especificados num padrão.

A estrutura e metodologia dos testes de conformidade OSI se constitui de um padrão definido a nível da ISO [7], [8], [9]. O escopo do padrão inclui definições de conceitos e termos, métodos de teste e proformas para o uso em protocolos padrão e relatórios de testes. Este padrão também define exigências para os fornecedores dos sistemas de testes e pacotes de testes executáveis, laboratórios de testes e seus clientes.

Conformidade no contexto do modelo OSI

Conformidade é uma das noções chave no contexto de sistemas abertos. Sendo conforme a um padrão internacional um sistema se torna aberto para todos os outros sistemas que utilizam o mesmo padrão, contudo, a noção de conformidade deve ser precisamente definida.

Conformidade no contexto da norma OSI diz respeito apenas à conformidade das implementações e sistemas reais para os padrões OSI, os quais especificam restrições aplicáveis. As restrições de conformidade num padrão podem ser: *restrições obrigatórias* (devem ser observadas em todos os casos), *restrições condicionais* (devem ser observadas se as condições ajustadas no padrão se aplicam) e *opções* (podem ser selecionadas para ajustar a implementação).

As declarações das restrições de conformidade num padrão podem ser iniciadas de duas formas: *positivamente* (iniciam com o que deve ser feito) ou *negativamente* (iniciam com o que não deve ser feito - proibições).

Pode-se, ainda, distinguir dois grupos de restrições de conformidade: as *restrições de conformidade estáticas* (que definem as capacidades mínimas permitidas de uma implementação, tais como o agrupamento de unidades funcionais e opções em classes de protocolos, ou o alcance dos valores que devem ser suportados para parâmetros específicos e temporizadores) e as *restrições de conformidade dinâmicas* (que definem o conjunto de comportamentos permitidos de uma implementação).

Um sistema ou implementação conforme deve satisfazer as restrições estáticas e as dinâmicas e ser consistente com as capacidades e opções indicadas no PICS (Protocol Implementation Conformance Statement). Sendo que o PICS é uma declaração, feita pelos fornecedores de uma implementação ou sistema OSI, especificando as capacidades e opções que devem ser implementadas e qualquer aspecto que deve ser omitido.

Os tipos de testes aplicados aos protocolos de comunicação

O objetivo principal dos testes de conformidade é verificar se a implementação que está sendo testada é conforme com a especificação do padrão. Limitações práticas tornam isso impossível de ser exaustivo e considerações econômicas podem restringir o teste ainda mais.

Todavia, existem quatro tipos de testes distintos, de acordo com o grau que eles proporcionam uma indicação de conformidade: os testes de interconexão básica, os testes de capacidade, os testes de comportamento e os testes de resolução de conformidade.

Os ***testes de interconexão básica*** proporcionam testes limitados de uma *IUT*[1] em relação aos principais aspectos num padrão, servem para estabelecer se existe conformidade suficiente para a interconexão ser possível, sem tentar executar através de testes. Estes testes são apropriados para detectar casos severos de não-conformidade; servir como um filtro preliminar antes de se aplicar testes mais apurados; dar uma primeira indicação de que uma implementação; a qual passou por um teste de conformidade completo num ambiente continua conforme num novo ambiente; determinar se uma implementação pode ser usada para comunicação com outras implementações conformes, por exemplo como uma troca de dados preliminar. Estes devem ser padronizados tanto como um pacote de testes muito pequeno ou como um subconjunto de um pacote de testes de conformidade.

Os ***testes de capacidade*** proporcionam testes limitados de cada uma das restrições de conformidade estáticas num padrão, para apurar qual capacidade de uma IUT pode ser observada e para checar quais dessas capacidades são válidas com respeito às restrições de conformidade estáticas e ao PICS. São apropriados para checar no máximo possível a consistência de um PICS com uma IUT; checar que as capacidades de uma IUT são consistentes com as restrições de conformidade estáticas; possibilitar uma seleção eficiente dos testes de comportamento a serem realizados para uma IUT particular; servir como uma base para assegurar conformidade quando realizados junto com os testes de comportamento. Eles devem ser padronizados dentro de um pacote de testes de conformidade. Eles podem tanto estar separados dentro de seus próprios grupos de teste ou fundidos com os testes de comportamento.

Os ***testes de comportamento*** testam uma implementação tão completamente quanto é prático, sobre o alcance total das restrições de conformidade dinâmicas especificadas num padrão. Desde que o número de combinações possíveis dos eventos e o tempo dos eventos é infinito, tais testes não podem ser exaustivos. Uma limitação adicional é que este tipo de teste é projetado para ser executado coletivamente num único ambiente de teste, tal que qualquer falha que seja difícil de ser detectada neste ambiente provavelmente será perdida, por esta razão é possível que uma implementação não-conforme passe por um pacote de testes de conformidade. Eles devem ser padronizados como o corpo de um pacote de testes de conformidade.

Os ***testes de resolução de conformidade*** proporcionam respostas, tão próximas das definitivas quanto possível, para a questão de uma implementação satisfazer restrições particulares, devido a problemas de exaustividade as respostas definitivas são atingidas às custas de testes de conformidade para um campo estreito.

A diferença entre os testes de comportamento e os testes de resolução de conformidade pode ser ilustrada pelo caso de um evento tal como o RESET. Os testes de comportamento devem incluir apenas uma seleção representativa das condições sobre as quais um RESET pode ocorrer e pode falhar para detectar um comportamento incorreto em outras circunstâncias. Os testes de resolução de conformidade seriam limitados às condições sobre as quais o comportamento incorreto já era suspeito de ocorrer, e confirmará ou não se a suspeita era correta. Esta classe de testes permite: obter respostas sim/não numa situação estritamente limitada e previamente identificada (por exemplo durante o desenvolvimento de uma implementação para checar se um aspecto particular foi implementado corretamente, ou durante o uso operacional, para investigar as causas dos problemas); identificar e oferecer

[1] *Uma implementação que está sendo testada é chamada de IUT (Implementation Under Test).*

resolução para deficiências em pacote de testes de conformidade. Os testes de resolução de conformidade não são padronizados.

Em adição aos testes de conformidade, outros tipos de testes têm sido propostos, incluindo: ***testes de interoperabilidade*** (para determinar se duas IUT's irão de fato inter-operar e se não, porque não); ***testes de desempenho*** (para medir as características de desempenho de uma IUT); ***testes de robustez*** (para determinar o quanto bem uma IUT recupera-se de várias condições de erro).

Pode existir alguma sobreposição entre estes tipos de testes e os testes de conformidade, onde houver isto então estes testes podem ser colocados como um subconjunto de um pacote de testes de conformidade completo.

3. GERAÇÃO DE SEQÜÊNCIAS DE TESTE X ESPECIFICAÇÕES FORMAIS

Como já foi dito neste documento, uma boa escolha das seqüências de teste está intimamente ligada à obtenção de um grau satisfatório de equillíbrio entre as duas propriedades desejáveis para o processo de teste: a ***eficiência*** e a ***eficácia***.

Nesta seção, apresentaremos alguns dos métodos presentes na literatura no que diz respeito à geração de seqüências de teste de conformidade. Como será visto, todos estes métodos são baseados na existência de uma especificação formal, da qual serão derivadas as seqüências, sendo que a distinção que se pode fazer entre estes métodos são, principalmente, as hipóteses adotadas para a identificação dos estados de uma implementação e a técnica de especificação utilizada para o protocolo.

No que diz respeito a este segundo ponto, podemos destacar os trabalhos realizados com base na especificação em máquinas de estado finito, assim como aqueles orientados à utilização das técnicas de descrição formal. Apresentaremos, na seqüência, alguns destes métodos.

Métodos baseados em máquinas de estado finitos

Uma técnica bastante utilizada para modelar um protocolo é um sistema de transição de estado. A especificação de um protocolo pode ser dividida em duas partes: a estrutura de dados e a estrutura de controle. Uma FSM é um sistema de transição de estados simples que é tipicamente usada para modelar a estrutura de controle dos protocolos. Devido a estas razões, técnicas baseadas em FSM's têm sido largamente utilizadas na geração de seqüências de teste.

Antes de se descrever os métodos, algumas definições precisam ser apresentadas:

FSM determinística. Uma FSM é dita determinística se sua saída e o próximo estado estão em função do seu estado atual e da entrada que é aplicada. Para uma FSM determinística não podem existir dois ramos deixando um único vértice com a mesma operação de entrada, se isto acontecer, tais ramos, sem levar em consideração as suas operações de saída, implicarão que uma mesma operação de entrada pode causar duas transições de estado diferentes iniciando no mesmo estado, o que caracteriza um comportamento não determinístico.

FSM completamente especificada. Uma FSM é dita completamente especificada se os conjuntos de entradas permitidas para cada estado são equivalentes; qualquer entrada é permitida em qualquer estado e gera uma saída permitida, em outras palavras uma FSM é completamente especificada se a partir de cada estado ela possui uma transição para cada símbolo de entrada. Caso contrário a FSM é dita parcialmente especificada.

FSM fortemente conectada. Uma FSM é dita fortemente conectada se para cada par de estados (s_i e s_j) existe um caminho de transições indo de s_i para s_j.

Segundo a literatura estes métodos podem ser divididos em quatro grupos: o método T (Transition Tour), o método D (Distinguishing Sequences), o método W (Characterizing Sequences) e o método U (Unique Input/Output Sequences).

Os quatro métodos assumem que a estrutura de controle de um protocolo é modelada como uma FSM determinística, completamente especificada e que a implementação é fortemente conectada, esta última hipótese corresponde a dizer que a especificação está livre de deadlocks e livelocks.

Apresenta-se, a seguir, uma breve descrição de cada um dos métodos citados acima:

O Método T (Transition Tour). Neste método uma seqüência de teste é gerada baseada na especificação da FSM, tal que, quando esta seqüência é aplicada à implementação da FSM qualquer transição de estado definida pela especificação é testada. É óbvio que uma seqüência de teste gerada pelo método T apenas verifica a existência de transições e não testa os estados finais destas transições porém, a grande vantagem deste método é a sua simplicidade quando comparado com os outros três métodos.

O Método D (Distinguishing Sequences). As seqüências geradas por este método (denominadas de seqüências DS) são definidas como um conjunto de entradas que gera um conjunto de saídas diferentes para cada estado da FSM. A seqüência de teste para a FSM a ser testada deve ser encontrada baseada na especificação. As principais desvantagens deste método são que em implementações típicas muito poucas FSM's possuem seqüências DS, e quando possuem essas seqüências são muito longas, tornando o uso deste método bastante limitado.

O Método W (Characterizing Sequences). Para as FSM's que não possuem seqüências DS, o método de seqüências de caracterização define seqüências DS parciais, cada uma das quais distingue um estado específico da FSM de um subconjunto dos estados restantes ao invés de distinguir este estado de todos os estados da FSM. O conjunto completo dessas seqüências de entrada para a FSM é chamado de conjunto característico **W** da FSM, este conjunto consiste de seqüências de entrada tal que os últimos símbolos de saída observados a partir da aplicação dessa seqüência (em uma ordem fixada) são diferentes para cada estado da FSM. Este método é mais aplicável a implementações típicas do que o método D.

O Método U (Unique Input/Output Sequences). Neste método uma seqüência de entrada de custo mínimo é encontrada para cada estado s_i da FSM, denotada como UIO(s_i), tal que a saída gerada por UIO(s_i) é única para o estado s_i, ou seja este comportamento de entrada/saída não é exibido por nenhum outro estado da FSM. Estas seqüências UIO são usadas para gerar uma seqüência de teste que visita cada estado de uma transição. A seqüência UIO para um estado s_i pode apenas verificar que a FSM estava no estado s_i antes da seqüência UIO ser aplicada e não identifica o estado dessa FSM caso ela esteja em um outro estado que não seja s_i. Formalmente uma seqüência UIO(s_i), é uma seqüência de entrada/saída especificada $S = (i_1/o_1)\ (i_2/o_2)\ \ldots\ (i_p/o_p)$ tal que não existe nenhum outro estado para o qual UIO(s_i) é uma seqüência entrada/saída especificada.

Uma descrição mais detalhada destes métodos pode ser encontrada em [11] e [12].

Métodos baseados em técnicas de descrição formal

O uso de técnicas de descrição formal para a especificação de protocolos de comunicação e mesmo outras classes de aplicações distribuídas tem suscitado a realização de trabalhos orientados à definição de métodos de geração de seqüências de teste a partir destas especificações.

Entretanto, o grande poder de expressão oferecido pelas técnicas de descrição formal como ESTELLE, LOTOS e SDL, se por um lado permitem ao programador um alto grau de abstração nos processos de síntese dos protocolos, aparecem como um obstáculo nos processos de análise do mesmo.

Sendo assim, como será visto a seguir, a aplicação de métodos de geração de seqüência de testes impõe a realização de simplificações nos modelos adotados para estas técnicas, de modo que, sem perder totalmente o seu poder de expressão, as especificações sejam representadas numa forma mais adequada à aplicação de um algoritmo de geração de seqüências de teste.

No que diz respeito à técnica ESTELLE, por exemplo, podemos destacar o trabalho realizado em [13], onde um modelo orientado ao teste (*TOF, Test Oriented Form*) é obtido a partir de uma especificação simplificada do protocolo.

Em [13], uma primeira simplificação imposta na especificação ESTELLE original é a obtenção de um único módulo representando o comportamento global da entidade de protocolo a ser testada (IUT). Esta simplificação é também definida em [14], que propõe a fusão de módulos como forma de obtenção de um módulo resultante único.

A eliminição de algumas construções do Pascal, tais como IF e FOR são requisitos para a obtenção da TOF definida em [13]. Ainda, as cláusulas PROVIDED são modificadas no sentido de incorporar as decisões a serem tomadas no contexto da construção IF. No caso da construção FOR, em geral, várias transições são definidas. Em [15], as transições tendo o mesmo comportamento observável são identificadas e agrupadas numa única transição, sendo que a cláusula PROVIDED resultante deverá reagrupar as condições de todas as cláusulas PROVIDED das transições resultantes.

Em [13] é proposta a eliminição da cláusula PRIORITY, sendo que a prioridade de uma transição em relação às demais é resolvida também pela adição de termos à cláusula PROVIDED.

No que diz respeito à técnica de descrição formal LOTOS, o princípio da geração de seqüências de teste é baseado na mesma classe de dificuldades encontradas no caso das especificações em ESTELLE.

Neste caso, porém, dado que LOTOS é uma técnica fundada em álgebra de processos, as transformações a serem consideradas, deverão ser realizadas sobre os operadores definidos na álgebra correspondente. Algumas das transformações definidas nos trabalhos em LOTOS são:

- transformação do operador sequencial num operador paralelo semanticamente equivalente;
- transformação do operador de sincronização total num operador paralelo generalizado;
- remoção de eventos internos escondidos (pelo operador *hide*, de LOTOS).

De forma compatível com os métodos orientados para ESTELLE, o objetivo destas simplificações é gerar, a partir destas especificações normalizadas, uma especificação orientada a testes, representada num modelo mais adequado. Um exemplo de tal modelo é o Chart, cuja definição formal pode ser verificada em [16]. Em [17] e [18] podem ser verificadas outras abordagens de geração de seqüências de teste a partir de LOTOS.

Em resumo, podemos considerar o processo de geração de seqüências de teste a partir de técnicas de descrição formal como sendo composto das seguintes etapas:

- aplicação de um conjunto de transformações à especificação original de modo a obter uma forma normalizada (simplificada);
- tradução da forma normalizada num modelo direcionado à geração das seqüências de teste;
- aplicação de um algoritmo de geração de seqüências de teste; a geração das seqüências de teste a partir de uma especificação em TOF, [13], por exemplo, é implementada por um algoritmo para FSM, adaptado, como aqueles apresentados na seção anterior.

4. METODOLOGIA DE GERAÇÃO DE SEQÜÊNCIA DE TESTES

Como foi mostrado na seção precedente, é possível encontrar na literatura um grande número de trabalhos evocando métodos de geração de seqüências de teste a partir de especificações formais, sejam elas representadas utilizando máquinas de estado ou técnicas de descrição formal como LOTOS e ESTELLE.

No que diz respeito às técnicas de descrição formal, e como descrito anteriormente, a totalidade dos métodos apresentados impõe a necessidade de simplificações preliminares nas especificações formais. Estas simplificações têm por objetivo eliminar parte do alto poder de expressão destas técnicas que, se por um lado, favorecem, em muito, a tarefa de especificação de sistemas complexos tais quais os protocolos de comunicação, introduzem dificuldades incontornáveis à aplicação de algoritmos de geração de seqüências de teste. A metodologia aqui apresentada, apesar de requerer, similarmente, simplificações na especificação original, propõe um escalonamento maior nestas simplificações, sendo que boa parte destas podem ser automatizadas, até que, pela aplicação de um método orientado às máquinas de estados finitos, é possível obter as seqüências de teste.

Intuitivamente, a metodologia proposta baseia-se na obtenção, a partir de uma especificação formal, de uma máquina de estados que represente o *comportamento observável*[2] do sistema a ser testado, no caso uma entidade de protocolos. Obtida a máquina de estados, o conjunto de seqüências de teste é obtido pela aplicação de um método de geração definido para esta categoria de modelos.

Sendo assim, podemos estruturar o método aqui proposto segundo as seguintes etapas:

Etapa I - Transformação da Especificação Formal

Nesta primeira etapa do método, a principal tarefa é a preparação da especificação formal para a etapa a seguir. Apesar de ser esta a única etapa não automatizada do processo de geração de testes, ela não impõe grandes dificuldades nem reduções significativas no poder de expressão da técnica de descrição formal utilizada, o que vem a ser, sem dúvida, uma grande vantagem deste método.

A tarefa de preparação da especificação formal consiste em traduzir, manualmente, a especificação formal descrita em ESTELLE (versão ISO) para uma nova especificação descrita em ESTELLE*. ESTELLE* é uma versão estendida de ESTELLE caracterizada por um maior poder de expressão graças à introdução de um novo mecanismo de comunicação (o *rendez-vous*) e outras modificações na semântica original de ESTELLE. Para maiores detalhes sobre esta versão, recomendamos [19] e [20].

---

[2] *Por comportamento observável entende-se a máquina de estados composta por um conjunto de transições, cujas etiquetas representem eventos observáveis, ou seja, eventos ocorrendo nas interfaces externas do sistema sob observação. No caso de uma entidade de protocolos, estes eventos representam, normalmente, primitivas de serviços (e/ou, em alguns casos, unidades de dados de protocolos – PDUs) trocadas entre a entidade e as demais componentes do sistema.*

A justificativa para a transformação de uma especificação ESTELLE para outra descrita em ESTELLE* será apresentada posteriormente.

Estas modificações a serem introduzidas apresentam-se em duas categorias: as modificações no mecanismo de comunicação de ESTELLE (filas FIFO) para o mecanismo introduzido na versão ESTELLE* (*rendez-vous*); as modificações estruturais na especificação ESTELLE.

A primeira classe de modificações resume-se a alterações nas transições da especificação em ESTELLE que apresentem cláusulas de recepção e/ou instruções de envio de interações. Um exemplo disto são as transições contendo cláusulas de recepção (WHEN), que se apresentam na forma descrita em (1) e que serão alteradas por substituição destas por cláusulas de sincronização em recepção como definidas em ESTELLE*, assumindo a forma mostrada em (2):

```
(1) TRANS
      WHEN ip.interação
        begin...end;
```

```
(2) TRANS
      ip?interação
        begin...end;
```

Para este tipo de transformação, as demais cláusulas (PROVIDED, por exemplo) são mantidas inalteradas, assim como as instruções contidas no bloco *begin ... end* da transição (com exceção da instrução OUTPUT de ESTELLE).

Um outro exemplo típico de modificação é aquele de transições contendo a instrução OUTPUT em seu bloco *begin ... end*, como mostrado em (3), que deverão também ser modificadas, de modo a assumir a forma mostrada em (4):

```
(3) TRANS
      ...
      begin
        ...
        OUTPUT ip.interação
      end;
```

```
(4) TRANS
      ip!interação
      begin
        ...
      end;
```

Com relação às modificações estruturais da especificação, a primeira das modificações a serem introduzidas na especificação consiste na transformação dos atributos ESTELLE ("*systemactivity*", "*systemprocess*", "*activity*" e "*process*") utilizados para definir o modo de paralelismo a ser imposto à implementação. Independente do atributo considerado na especificação, sua configuração final deverá ser caracterizada pela associação do atributo "*systemactivity*" à especificação e "*activity*" aos demais módulos que a compõem. Esta, apesar de ser uma restrição imposta para o uso da técnica ESTELLE*, permite cobrir, sob o ponto de vista do paralelismo, uma grande parcela dos possíveis comportamentos do sistema especificado. Outra modificação importante é a introdução (ou a substituição, se for o caso) dos "*módulos de ambiente*" que servirão de interface externa ao(s) módulo(s) que descrevem o comportamento da entidade de protocolos, cujas implementações serão objeto de teste de conformidade. A definição destes módulos é necessária, ao contrário de outras técnicas, uma vez que uma especificação ESTELLE deve ser escrita na forma de um "*mundo fechado*", desprovido portanto de interfaces com o "*exterior*". Por outro lado, do ponto de vista do comportamento, estes módulos de ambiente, apresentam-se relativamente simples, dado que eles servem de acessório para as comunicações entre o módulo que representa a entidade de protocolo sob teste e o ambiente com o qual a implementação vai interagir. Os corpos representando o comportamento destes módulos de ambiente vão, na maior parte dos casos, ser caracterizados pela existência de um único estado global, a partir do qual (e para o qual) serão disparadas todas as transições. Estas, por sua vez, serão caracterizadas, na sua totalidade, por cláusulas de sincronização (*rendez-vous*) em emissão ou em recepção, sendo

que as interações vão representar as mensagens trocadas entre a entidade de protocolos e o seu ambiente: as primitivas de serviço e as unidades de dados de protocolo (PDUs). As transições devem, então, representar todas as possíveis interações de comunicação entre a entidade de protocolo e o seu ambiente, ou seja, todas as trocas de unidades de dados de protocolos e ou primitivas de serviço que irão ocorrer durante o funcionamento do protocolo.

A parte ação das transições (bloco *begin ... end*) poderá ser totalmente vazia ou caracterizada por funções ou procedimentos de cálculo dos parâmetros das interações.

Com relação à declaração da máquina de estados propriamente dita, dado que esta é caracterizada pela existência de um estado único, não existe necessidade da utilização da declaração STATE, que permite definir os estados globais de cada corpo de módulo. A limitação no que diz respeito às possíveis interações podendo ocorrer entre o módulo que representa a entidade de protocolo e os demais módulos (ordenação de primitivas de serviço e PDUs) será controlada unicamente pela máquina (pelas máquinas) de estado representando a entidade (uma vez que esta é caracterizada pela existência de estados globais e outras componentes - variáveis, cláusulas PROVIDED, etc... - que permitem adicionar restrições de comportamento a esta máquina) e pelo uso do mecanismo de comunicação por *rendez-vous*.

Etapa II - Obtenção da Máquina de Estados

A máquina de estados que representa o comportamento observável da entidade de protocolos vai ser o objeto de aplicação da técnica de geração de seqüências de teste, técnica esta orientada ao modelo máquinas de estado finito.

A obtenção desta máquina de estados, portanto, requer o processamento da especificação formal descrita em ESTELLE de modo a obter todas as possibilidades de execução do módulo (ou dos módulos) representando a entidade de protocolos, na forma de um conjunto de estados e de transições etiquetadas pelas possíveis interações na forma de primitivas de serviço e de PDUs, interações estas que comporão as seqüências de teste que serão geradas posteriormente.

Por outro lado, a máquina de estados gerada deverá "*esconder*" a totalidade do comportamento interno da especificação da entidade de protocolos, traduzida em ESTELLE por transições expontâneas nos módulos (transições sem cláusulas ou instruções de comunicação) ou transições associadas a comunicações internas (entre os módulos compondo a especificação da entidade de protocolo).

Para isto, vamos utilizar as facilidades de verificação existentes no ambiente ESTIM (*ESTELLE SimulaTor based on an Interpretative Machine*) desenvolvido no LAAS-CNRS (França), [21], e que permite obter o grafo de alcançabilidade de uma especificação de protocolo descrito em ESTELLE*. Esta é a razão pela qual a transformação de uma especificação formal em ESTELLE para ESTELLE* é uma etapa indispensável dentro da metodologia proposta.

O grafo construído pela ferramenta de verificação ESTIM representa todas as possibilidades de execução da especificação formal submetida, sendo que este vai apresentar normalmente duas classes de transições representando o comportamento: transições etiquetadas por interações externas, que são aquelas associadas à troca de interações entre o módulo (ou módulos) representando a parte da especificação que está sob análise; transições etiquetadas por *null*, que representam aquelas transições relacionadas ao comportamento "interno" do módulo sob análise.

Para a obtenção da máquina de estados representando o comportamento observável da entidade de protocolo a ser testada, é necessário ainda, eliminar a segunda classe de transições descrita acima. Esta eliminação é realizada de forma computacional, pela aplicação de relações de equivalências entre autômatos, sendo que, para isto, adota-se a equivalência de rastro. Esta

equivalência permite eliminar, do grafo gerado por ESTIM, todas as transições internas, gerando um autômato (dito autômato quociente) caracterizado unicamente pelas transições etiquetadas pelas interações externas.

A automatização do método de redução do grafo por equivalência de rastro está presente num módulo da ferramenta PIPN (*Prolog Interpreted Petri Nets*), também desenvolvida no LAAS-CNRS, [22], com o qual o ambiente ESTIM é interfaceado.

Etapa III - Transformação da Máquina de Estados

Nesta etapa o arquivo de saída do grafo de acessibilidade gerado pela ferramenta ESTIM é transformado para a forma de uma FSM intermediária, contendo as seguintes informações do grafo original: o estado inicial da transição (ou ramo), a interação de entrada desse estado, a interação de saída desse estado e o estado final da transição. Quando um estado não possui a interação de entrada ou a interação de saída, então ela deve ser preenchida com a interação *null*, representando uma entrada ou saída nula.

Em seguida, a FSM intermediária sofre uma nova transformação, cujo objetivo é eliminar os estados artificiais introduzidos durante a etapa de especificação em ESTELLE*. A eliminação dos estados artificiais é realizada através da fusão de transições ($a\,f\,b$, sendo a e b duas transições), da seguinte maneira:

1. é verificado se o estado de chegada da transição a é igual ao estado de saída de b, com esta regra tem-se que o estado de saída de ($a\,f\,b$) é igual ao estado de saída da transição a e o estado de chegada de ($a\,f\,b$) é igual ao estado de chegada de b;
2. a apresenta um *label não nulo/nulo* e b apresenta *label nulo/não nulo*, com esta regra tem-se que o label de ($a\,f\,b$) é igual ao label de a unido com o label de b;
3. b é a única transição de saída do seu estado de saída.

O resultado desta fusão é a FSM que representa o fluxo de controle do protocolo especificado.

Etapa IV - Geração das Seqüências de Teste

O método de geração de seqüências de teste implementado detecta todas as faltas se o número de estados na implementação for igual ao da especificação. É assumido que a estrutura de controle do protocolo é modelada como uma FSM determinística, mínima e completamente especificada. A implementação é vista como uma caixa preta (*black box*) com uma porta de entrada e uma porta de saída, possuindo duas interfaces, uma interface com o usuário (ou com o protocolo da camada superior) e uma interface com o protocolo da camada inferior. O método utiliza a *"Completeness Assumption"* que diz o seguinte: a entidade de protocolo ignora uma entrada aplicada num estado para o qual a saída e o comportamento do próximo estado não estão especificados no *Core Behaviour* (comportamento desejável do sistema, este comportamento pode não cobrir todas as combinações possíveis de entrada/saída).

Para a FSM mostrada na figura 1, por exemplo, o conjunto dos ramos desejáveis é: {(01), (12), (24), (40), (61), (56), (35), (13), (00), (10), (20), (30), (40), (50), (60)}.

O modelo de FSM utilizado é aquele apresentado na seção anterior, obedecendo todas as restrições impostas a este modelo.

Para cada ramo (transição de estado) $e \in E$ são definidas três funções: *Head*(e), *Tail*(e) e *Label*(e), onde *Head*(e) denota o estado inicial para e, *Tail*(e) denota o estado final de e, e *Label*(e) denota a transição e.

Neste método uma seqüência é definida como uma série de pares entrada/saída. Em termos formais uma seqüência S é dada como: $S = (i_1/o_1)\ (i_2/o_2)\ \ldots\ (i_k/o_k)$ onde $i_k \in$ I e $o_k \in$ O com p sendo um inteiro positivo variando de 1 até k.

A metodologia descrita tem como idéia chave o conceito da assinatura única, chamada de seqüência UIO (*Unique Input/Output Sequence*), descrita na seção anterior, para cada estado da especificação do protocolo, por exemplo, a seqüência UIO para o estado 6 da figura 1 é *a/f c/d*. A escolha deste método de geração é justificada pelo fato de que quase todas as FSM's possuem este tipo de seqüência e o comprimento dessas seqüências é menor quando comparadas com os outros métodos de geração de seqüências de testes a partir de FSM's.

Para um estado s_i numa FSM que não possue uma seqüência UIO é usada uma assinatura que distingue s_i dos outros estados restantes. Este conceito está apresentado a seguir:

O processo de construção dessa assinatura é o seguinte: para cada estado s_j, pertencente ao conjunto de estados $\{s_1, s_2, \ldots s_n\}$ com n elementos, existe uma seqüência de entrada/saída $IO(i,j)$ (onde $i \neq j$) com comprimento inferior a n que pode distinguir s_i de s_j . A primeira parte da assinatura para o estado s_i é uma seqüência de entrada $IO(i,1)$ que distingue s_i de s_1 e leva a máquina para o estado $s_{i,1}$. Então o menor caminho de $s_{i,1}$ para s_i é escolhido para levar a máquina de volta a s_i, este caminho sendo chamado de seqüência de transferência $T(1)$. O procedimento é aplicado de novo para distinguir s_i de todos os outros estados. A seguinte seqüência é usada como assinatura para os estados:

$$LD(s_i) = IO(i,1)@\overset{n}{\underset{k=2}{\#}} T_i(k-1)@IO(i,k) \text{ onde } k \neq 1 \text{ para } i \neq 1$$

ou

$$LD(s_i) = IO(1,2)@\overset{n}{\underset{k=3}{\#}} T_1(k-1)@IO(1,k) \text{ para } i = 1$$

Onde @ e # são usados para representar a concatenação de seqüências. O limite superior para o comprimento dessa assinatura é $2n^2$. Se algum estado não possuir uma seqüência UIO com comprimento inferior a $2n^2$ então a assinatura $LD(s_i)$ é usada.

O algoritmo para a geração das seqüências UIO executa os seguintes passos:

1 : para cada estado, são calculadas todas as seqüências de *entrada/saída* e é feita uma verificação para ver se cada uma dessas seqüências é única;

2 : se nenhuma seqüência única de comprimento 1 (um) for encontrada, então o mesmo procedimento é repetido para todas as seqüências de comprimento 2;

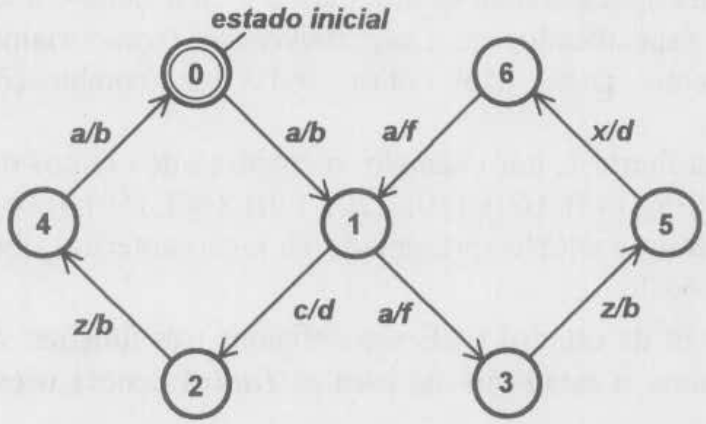

***Figura 1** - Máquina de Estados Finitos (FSM) exemplo.*

3 : repetir este procedimento para seqüências maiores até uma seqüência UIO ser encontrada ou o comprimento das seqüências exceder $2n^2$.

As seqüências de teste geradas pelo algoritmo descrito devem verificar se cada estado e cada ramo (ou transição de estado) da especificação existe na implementação e se cada um desses ramos possui um *label* (*i/o*) correto, para tal as seqüências de teste devem passar por todos os ramos. Com isso, para cada ramo a seqüência de teste executa os seguintes passos:

1 : leva a implementação para o seu estado inicial;
2 : aplica uma entrada para causar a transição de estado neste ramo;
3 : aplica a seqüência UIO para o estado final.

Após ter visitado cada ramo, o teste deve levar a implementação para o seu estado inicial através da entrada de *reset* (*ri*).

O algoritmo para geração de seqüências de teste executa os seguintes passos:

1 : a partir do estado inicial calcular os menores caminhos para todos os estados usando o *Algoritmo de Dijkstra's para o caminho mais curto* [23];
2 : gerar um roteiro de todos os ramos. O roteiro de ramo *e* exige uma seqüência de teste, denominada sub-seqüência: $TE(e) = P(Head(e))@Label(e)@UIO(Tail(e))$

Uma Ferramenta para a Geração de Seqüências de Teste

Uma decisão tomada desde o início dos trabalhos de definição da metodologia foi a de desenvolver ferramentas que suportassem totalmente ou grande parte desta. Ainda, um requisito importante era o de que a(s) ferramenta(s) deveria(m) estar disponível(is) num ambiente de grande difusão no meio acadêmico.

Por esta razão, duas ferramentas de suporte foram desenvolvidas sobre o sistema operacional UNIX, que acompanha as estações de trabalho, muito utilizadas atualmente nas universidades e outros centros de pesquisa.

A primeira ferramenta, ***transf***, implementa a etapa III da metodologia proposta, que consiste na transformação da máquina de estados gerada por ESTIM resultando na FSM a partir da qual serão geradas as seqüências de teste.

A segunda ferramenta, ***seqt***, implementa efetivamente o método de geração de seqüências de teste, segundo o método das seqüências UIO. Um aspecto interessante desta ferramenta é a possibilidade, segundo a opção do usuário, de obter resultados intermediários da etapa IV, o que dá à ferramenta um caracter didático. A figura 2 apresenta a tela de diálogo da ferramenta ***seqt***, onde se pode verificar as opções de saídas oferecidas ao usuário.

5. EXEMPLO DE APLICAÇÃO

Para este exemplo o protocolo de transporte simplificado proposto em [24] foi especificado em ESTELLE*. A escolha deste exemplo se justifica pelo fato de já ter sido previamente utilizado para ilustrar a aplicação de uma metodologia de geração de seqüências de teste de conformidade, o que permite um eventual estudo comparativo das duas técnicas.

O protocolo de transporte simplificado é baseado nas seguintes primitivas de serviço:

- ***TCONreq*** - solicitação de estabelecimento de conexão de transporte;
- ***TCONind*** - indicação de estabelecimento de conexão de transporte;
- ***TCONresp*** - resposta a indicação de estabelecimento de conexão de transporte;
- ***TCONconf*** - confirmação do estabelecimento de conexão de transporte;

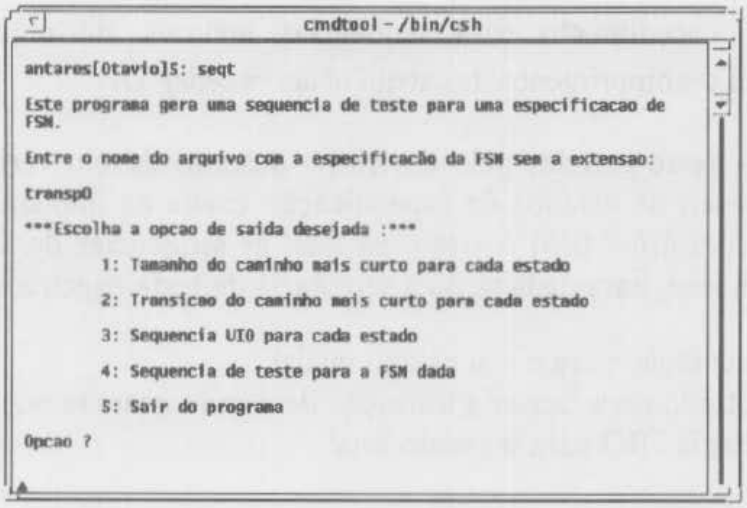

Figura 2 - Tela de apresentação da ferramenta ***seqt****.*

- ***TDATAreq*** - solicitação de estabelecimento de conexão de dados;
- ***TDATAind*** - indicação de estabelecimento de conexão de dados;
- ***TDISreq*** - solicitação de cancelamento da conexão de transporte;
- ***TDISind*** - indicação de cancelamento da conexão de transporte.

As unidades de dados de protocolos trocadas entre as entidades são as seguintes:

- *CR* - Connect Request
- *CC1* - Connect Confirm (negativo)
- *CC2* - Connect Confirm (positivo)
- *DT, AK* - Data, Acknowledged
- *DR* - Disconnect Request
- *DC* - Disconnect Confirm

Podemos descrever, informalmente, o funcionamento do protocolo, segundo as seguintes etapas:

I. Estabelecimento da Conexão de Transporte

- ***iniciada localmente:***

1. usuário emite a primitiva ***TCONreq***;
2. entidade envia ***PDU-CR*** à entidade par e aguarda confirmação;
3. se a PDU recebida é positiva (*CC2*), a entidade encaminha confirmação ao usuário e a conexão é considerada aberta (OPEN);
4. se a PDU é negativa (*CC1*), a entidade encaminha uma primitiva de serviço ***TDISind*** ao usuário e envia a ***PDU-DR*** à entidade par. A conexão é considerada fechada (CLOSED).

- ***iniciada remotamente (pela entidade par):***

1. entidade recebe ***PDU-CR*** e emite primitiva ***TCONind*** ao usuário, ficando a espera de resposta;
2. usuário emite resposta positiva (primitiva ***TCONresp***), entidade emite ***PDU-CC2*** considerando a conexão estabelecida (OPEN);

3. usuário emite resposta negativa (primitiva ***TDISreq***), entidade emite ***PDU-DR*** e aguarda resposta. Em seguida, a entidade encaminha uma primitiva de serviço ***TDISconf*** ao usuário e envia a ***PDU-DR*** à entidade par. A conexão é considerada fechada (CLOSED).

II. FASE DE TRANSFERÊNCIA DE DADOS

- ***emissão do dado:***

1. entidade recebe primitiva ***TDATAreq*** e emite o dado; utilizando ***PDU-DT***;
2. entidade recebe reconhecimento do dado; utilizando ***PDU-AK***.

- ***recepção do dado:***

1. entidade recebe ***PDU-DT***;
2. entidade emite dado ao usuário via primitiva ***TDATAind***;
3. entidade retorna reconhecimento à entidade par.

III. TERMINAÇÃO DA CONEXÃO DE TRANSPORTE

- ***iniciada localmente (usuário):***

1. entidade recebe pedido de desconexão do usuário (***TDISreq***);
2. entidade emite ***PDU-DR*** e fica a espera de resposta (da entidade par);
3. recebida a ***PDU-DC***, entidade envia confirmação de desconexão (primitiva ***TDISconf***) e retorna ao estado inicial (CLOSED).

- ***iniciada localmente (pela entidade de transporte):***

1. entidade emite indicação de terminação (***TDISind***) e ***PDU-DR***, ficando a espera de resposta;
2. recebida a ***PDU-DC***, entidade envia confirmação de desconexão (primitiva ***TDISconf***) e retorna ao estado inicial (CLOSED);
3. a entidade encaminha uma primitiva de serviço ***TDISconf*** ao usuário e envia a ***PDU-DR*** à entidade par. A conexão é considerada fechada (CLOSED).

- ***iniciada remotamente (pela entidade par):***

1. entidade recebe ***PDU-DR***;
2. envia indicação de desconexão (***TDISind***) e resposta ***PDU-DC***.

A partir da descrição informal do protocolo, foi realizada a especificação formal diretamente em ESTELLE*, cuja arquitetura é ilustrada pela figura 3. Ela é composta dos módulos USER_A, T_ENT e NETWORK que representam, respectivamente, o usuário da camada de transporte,a entidade de transporte e o serviço de comunicação.

O módulo USER_A comunica-se com o módulo T_ENT através do canal CH1 que interliga os pontos de interação denominados TSAP (tanto no módulo USER_A quanto em T_ENT). As interações que podem ser trocadas entre estes módulos são: TCONreq, TCONresp, TDISreq, TDISresp, TDATAreq e NULL, num sentido e TCONind, TCONconf, TDISind TDISconf e TDATAind, no outro.

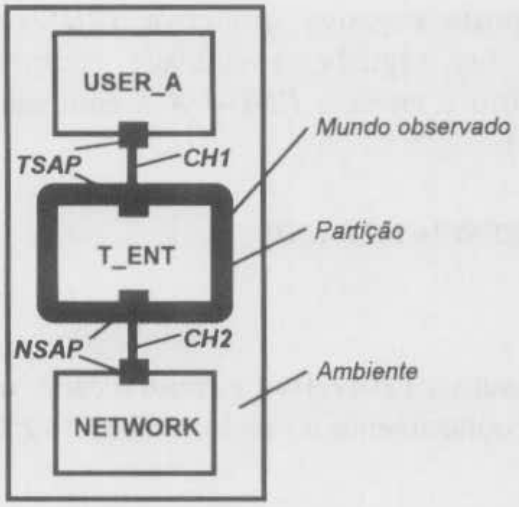

***Figura 3** - Arquitetura da especificação formal em ESTELLE* do protocolo de Transporte.*

O canal CH2, que interliga os módulos NETWORK e T_ENT através dos pontos de interação NSAP, permitindo representar a troca de PDU's entre as entidades de transporte. As interações circulando neste canal são: CR, CC, DR, DC, DT, CC1 e CC2 em ambos os sentidos.

Os corpos de módulos de USER_A e NETWORK são definidos de modo a representar as comunicações externas nas interfaces superior e inferior do módulo T_ENT. O corpo deste, por sua vez, descreve o comportamento do protocolo de transporte como apresentado na especificação informal.

A próxima etapa consiste na obtenção do autômato que descreve o comportamento observável do protocolo de transporte.

Para isto, ativa-se a ferramenta ESTIM, indicando o nome do arquivo que contém a especificação ESTELLE*. Terminada a inicialização, deve ser ativada a facilidade de geração do grafo de alcançabilidade (através do comando *"graph"*).

Em seguida, para que se possa lançar a geração do grafo, é necessário informar as fronteiras que servirão de referência para distinguir eventos observáveis e eventos invisíveis - é a definição de partição. A escolha é feita de modo que apenas o módulo T_ENT permaneça no interior da partição, os demais módulos vindo fazer parte do ambiente. A figura 3 ilustra esta escolha.

Sendo assim, a ferramenta vai considerar como observáveis, todos os eventos que estejam associados a trocas de interações nos canais CH1 e CH2. Qualquer outro evento será considerado invisível e será, portanto, etiquetado por *"nil"*.

Lançada a geração de grafo, o resultado obtido deverá sofrer ainda o processo de redução por equivalência de rastro. Isto é feito invocando-se o comando *"proj -l"* de dentro do ambiente ESTIM.

O autômato obtido representa, finalmente, o comportamento observável da entidade de transporte, sobre o qual serão aplicados as demais etapas da metodologia proposta.

O resultado foi aplicado à ferramenta ***transf*** para se obter a forma da FSM de entrada da ferramenta *seqt* (geração de seqüências de teste). Esta FSM foi aplicada para a ferramenta *seqt* de onde se obteve os resultados mostrados abaixo. Para facilitar a visualização dos resultados foi montada a FSM, apresentada na figura 4.

A seguir, são apresentadas as equivalências entre os estados da figura 4 e aqueles definidos na descrição informal do protocolo:

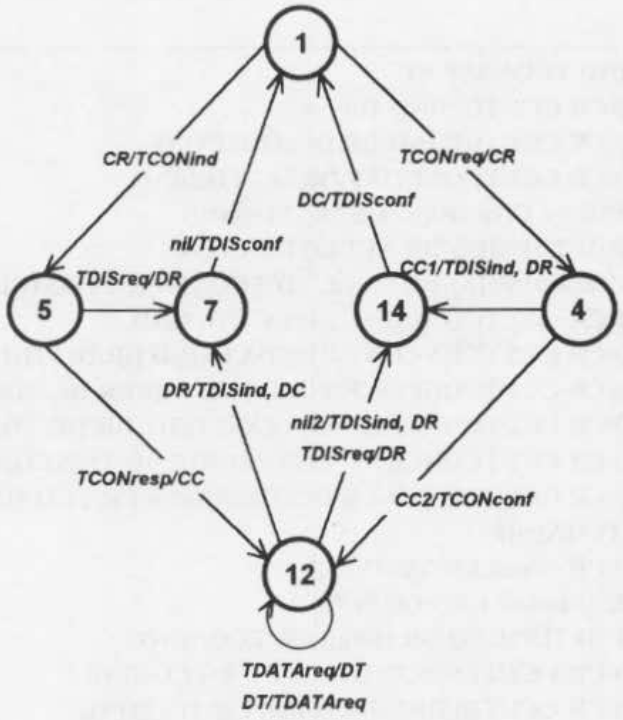

***Figura 4** - Máquina de Estados modelando o comportamento do protocolo de Transporte.*

Estado	Significado
1	closed
4	wait_for_CC
5	wait_for_TCONresp
7	closing
12	open
14	wait_for_DC

Ativando-se a ferramenta ***seqt***, pode-se, então, obter os seguinte resultados a partir da máquina de estados ilustrada pela figura 4:

Caminhos mais curtos da FSM (Algoritmo de Dijkstra):

Estado	Tamanho do caminho mais curto	Transição do caminho mais curto
1	0	ri/null
4	1	TCONREQ/CR
5	1	CR/TCONIND
7	2	CR/TCONIND TDISREQ/DR
12	2	TCONREQ/CR CC2/TCONCONF
14	2	TCONREQ/CR CC1/TDISIND-DR

Seqüências UIO:

Estado	Seqüência UIO
1	CR/TCONIND
4	CC1/TDISIND-DR
5	TCONRSP/CC
7	NULL/TDISCONF
12	DR/DC-TDISIND
14	DC/TDISCONF

Seqüências de teste:

ri/null CR/TCONIND TCONRSP/CC
ri/null TCONREQ/CR CC1/TDISIND-DR
ri/null TCONREQ/CR CC1/TDISIND-DR DC/TDISCONF
ri/null TCONREQ/CR CC2/TCONCONF DR/DC-TDISIND
ri/null CR/TCONIND TCONRSP/CC DR/DC-TDISIND
ri/null CR/TCONIND TDISREQ/DR NULL/TDISCONF
ri/null CR/TCONIND TDISREQ/DR NULL/TDISCONF CR/TCONIND
ri/null TCONREQ/CR CC2/TCONCONF DR/DC-TDISIND
ri/null TCONREQ/CR CC2/TCONCONF DT/TDATAIND DR/DC-TDISIND
ri/null TCONREQ/CR CC2/TCONCONF NULL/TDISIND-DR DC/TDISCONF
ri/null TCONREQ/CR CC2/TCONCONF TDATAREQ/DT DR/DC-TDISIND
ri/null TCONREQ/CR CC2/TCONCONF TDISREQ/DR DC/TDISCONF
ri/null TCONREQ/CR CC1/TDISIND-DR DC/TDISCONF CR/TCONIND
ri/null ri/null CR/TCONIND
ri/null TCONREQ/CR ri/null CR/TCONIND
ri/null CR/TCONIND ri/null CR/TCONIND
ri/null CR/TCONIND TDISREQ/DR ri/null CR/TCONIND
ri/null TCONREQ/CR CC2/TCONCONF ri/null CR/TCONIND
ri/null TCONREQ/CR CC1/TDISIND-DR ri/null CR/TCONIND

Por comparação, foi possível verificar a compatibilidade das seqüências geradas com aquelas obtidas em [24].

6. CONCLUSÕES

O artigo apresentou os resultados dos estudos realizados objetivando o desenvolvimento na área de teste de conformidade de protocolos de comunicação. O destaque a ser dado sobre os resultados obtidos está no estudo de métodos de geração de seqüências de teste de conformidade existentes na literatura, na definição de uma metodologia original baseada na utilização da técnica de descrição formal ESTELLE, e no desenvolvimento de ferramentas suportando tal metodologia.

Um exemplo ilustrativo permitiu demonstrar o uso da metodologia proposta, assim como de verificar sua eficiência, pela comparação com resultados obtidos, com uso de outro método, sobre o mesmo exemplo.

Naturalmente, os trabalhos não esgotam-se com estes resultados, podendo-se vislumbrar diversos outros estudos a serem realizados. Um aspecto importante a ser explorado, a partir deste trabalho, é a aplicação da metodologia e das ferramentas sobre exemplos mais representativos e mais complexos de protocolos de comunicação. Um outro trabalho importante seria o desenvolvimento de uma ferramenta permitindo automatizar a primeira etapa da metodologia proposta, ou seja, a transformação de especificações formais de ESTELLE para ESTELLE*.

Finalmente, um outro resultado esperado como continuidade a este trabalho é a integração completa de todas estas ferramentas, incluindo a ferramenta ESTIM, num mesmo ambiente de auxílio ao desenvolvimento de protocolos de comunicação a partir de especificações formais.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] H. Zimmermann. *"OSI Reference model: the ISO model of architecture for Open Systems Interconnection"*, IEEE Transactions on Communications, Vol. COM-28, nº 4, Abril 1980, pp. 425-432.

[2] A. A. S. Danthine. *"Protocol representation with Finite-State Models"*, IEEE Transactions on Communications, Vol. COM-28, n. 4, April (1980), pp. 632-643.

[3] M. Diaz. *"Modeling and Analysis of Communication and Cooperation Protocols using Petri Net Based Models"*, Computer Networks & ISDN Systems, Vol. 6 (1982), pp. 419-441.

[4] ISO IS 9074. *"ESTELLE - A Formal Description Technique Based on a Extended State Transition Model"*, International Standardization Organization, Novembro (1988).

[5] ISO IS 8807. *"LOTOS, A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour"*, Novembro 1988.

[6] CCITT/SGXI/WP3-1. "SDL, Specification and Description Language", CCITT Recommendations Z100-Z104, 1988.

[7] ISO/IEC JCT 1/ SC 21. *"Information Processing Systems - OSI Conformance Testing Methodology and Framework"*, Parts 1-2, Novembro (1988).

[8] ISO/IEC JCT 1/ SC 21. *"Information Processing Systems - OSI Conformance Testing Methodology and Framework"*, Part 3, February (1989).

[9] ISO/IEC JCT 1/ SC 21. *"Information Processing Systems - OSI Conformance Testing Methodology and Framework"*, Parts 4-5, March (1989).

[10] R. J. Linn. *"Conformance Testing for OSI Protocols"*, Computer Networks and ISDN Systems 18 (1989/90) 203-219.

[11] B. S. Bosik, M. U. Uyar. *"Finite state machine based formal methods in protocol conformance testing: from theory to implementation"*, Computer Network and ISDN Systems 22 (1991), pp. 7-33.

[12] D. P. Sidhu, T.-K. Leung. *"Formal Methods for Protocol Testing: A Detailed Study"*, IEEE Transactions on Software Engineering, Vol. 15, n. 4, Abril (1989), pp. 413-426.

[13] W. Chun, P. D. Amer. *"Test Case Generation for Protocols Specified in ESTELLE"*, Proceedings of the IFIP International Conference on Formal Description Techniques - FORTE'90, Madrid (Espanha), Novembro (1990), pp. 197-209.

[14] B. Forghani, B. Sarikaya. *"Automatic Dynamic Behaviour Generation in TTCN from ESTELLE Specifications"*, Proceedings of X IFIP Conference on Protocol Specification, Testing and Verification (1990) 125-139.

[15] M. Phalippou, R. Groz. *"From ESTELLE specifications to industrial test suite, using an empirical approach"*, Proceedings of the IFIP International Conference on Formal Description Techniques - FORTE'90, Madrid (Espanha), Novembro (1990), pp. 179-196.

[16] P. Tripathy, B. Sarikaya. *"Test Generation from LOTOS Specifications"*, IEEE Transactions on Computers, Vol. 40, n. 4, Abril (1991), pp. 543-552.

[17] D. H. Pitt, D. Freestone. *"The Derivation of Conformance Tests from LOTOS Specifications"*, IEEE Transactions on Software Engineering, Vol. 16, n. 12, December (1990), pp. 1337-1343.

[18] K. Naik, B. Sarikaya. *"Testing Communication Protocols"*, IEEE Software, Janeiro (1992), pp. 27-36.

[19] J.P. Courtiat. *"Contribution à la Description Formelle de Protocoles"* Thèse de Doctorat d'Etat. Université Paul Sabatier, Toulose, 1987.

[20] J.P. Courtiat et alli. *"ESTELLE*: An ISO language for distributed algorithms and protocols"*, Technology and Science of Informatics. vol. 6, n° 5, pp 311-324, 1987.

[21] P. de Saqui-Sannes. *"Prototypage d'un Environnement de Validation de Protocoles: Application à l'Approche ESTELLE"*, Tese de Doutorado, Toulouse, Abril, 1990.

[22] J. C. Lloret. *"Réseaux Predicat/Transition Etiquétés pour la Modélisation et la Vérification de Systèmes Informatiques Répartis"* Tese de Doutorado, Toulouse, 1990.

[23] L. O. Rodrigues da Silva. *"Uma Metodologia de Geração de Seqüências de Teste a partir de Especificações descritas na Técnica de Descrição Formal ESTELLE"*, Dissertação de Mestrado, CPGEEL-UFSC, Florianópolis-SC, Agosto de 1994.

[24] K. Sabnani, A. Dahbura. *"A Protocol Test Generation Procedure"*. Computer Networks and ISDN Systems 15 (1988). 285-297.