Geração Automática de Casos de Teste de Conformidade para Software de Aplicações em Protocolos de Comunicação

Ana Silvia M. S. do Amaral
Laboratório Associado de Computação e Matemática Aplicada (LAC)
Instituto Nacional de Pesquisas Espaciais (INPE)
BRAZIL
anasil@lac.inpe.br

N. L. Vijaykumar
Laboratório Associado de Computação e Matemátca Aplicada (LAC)
Instituto Nacional de Pesquisas Espaciais (INPE)
BRAZIL
Coastal and Marine Resources Centre (CMRC)
University College Cork(UCC)
University College Cork (UCC)
IRELAND
n.vijaykumar@ucc.ie

Eliane Martins
Instituto de Computação Unicamp
BRAZIL
eliane@ic.unicamp.br

Resumo

Este trabalho apresenta uma metodologia para geração automática de casos de teste de conformidade para software de aplicação em protocolos de comunicação. A especificação se baseia na técnica Statechats e a metodologia consiste em converter a especificação Statecharts para um Máquina Finita de Estados Estendida (EFSM). A partir daí, faz-se uso da ferramenta Condado para a geração automática de casos de teste que permite tratar os aspectos controle e dados de maneira unificada.

Abstract

This work presents a methodology for automatic conformance test case generating for communicating protocol application software. The specification is based on the technique Statecharts and the methodology consists in converting the Statecharts specification to the Extended Finite State Machine (EFSM). After this, the Condado tool is used for the automatic test case generating enabling both the control and data aspects on an unified basis.

1. Introdução

Ao contrário de sistemas transformacionais, alguns sistemas não podem ser descritos simplesmente baseados em certas entradas e suas respectivas saídas resultantes. São os chamados sistemas reativos, os quais exigem uma análise dessas relações de entrada e saída ao longo do tempo. Um exemplo simples é uma lâmpada, ou seja, é importante a ordem de chaveamento das posições ligada/desligada. Sistemas reativos, ao contrário de sistemas transformacionais, são sistemas que reagem continuamente a estímulos internos e externos. Alguns exemplos de sistemas reativos: telefones, automóveis, redes de comunicação, semáforos, sistemas embarcados de tempo real, sistemas aviônicos, etc. Devido às suas características especiais de responder aos estímulos, os sistemas reativos necessitam de técnicas para representá-los. Entre algumas técnicas, podem ser mencionadas: diagramas de estado ou máquina finita de Estados (FSM), redes de filas, redes de Petri e Statecharts. Estas técnicas possuem a característica de definir, precisamente, a ordem temporal de suas interações. Isso explica o fato de Máquinas Finitas de Estados (FSM) estarem sendo frequentemente utilizadas na especificação de tais sistemas em diversas áreas e, mais recentemente, em protocolos de comunicação. Protocolos de comunicação são um conjunto de regras definidas que regem a troca ordenada de mensagens entre entidades de comunicação. Consequentemente, muitos trabalhos relacionados à área de testes de protocolos têm sido baseados em modelos de Máquinas Finitas de Estados (FSM). A técnica de especificação Statecharts pode ser considerada como Máquinas Finitas de Estados Estendidas (EFSM) que suportam estrutura hierárquica e concorrente de estados e um mecanismo de comunicação entre componentes através de

eventos. Nossa abordagem nesse trabalho é a geração de casos de testes a partir de Máquinas Finitas de Estados Estendidas (EFSM), as quais estendem as FSM através da inclusão de variáveis de contexto e parâmetros de interações. Existe um trabalho, desenvolvido no LAC/INPE (Laboratório de Computação e Matemática Aplicada), sobre modelagem de desempenho baseada na especificação Statecharts [21] e [22], onde a cadeia de Markov é gerada a partir da especificação do sistema em Statecharts. Na verdade, o modelo Markov gerado, tem uma correspondência com a Máquina Finita de Estados (FSM). A proposta deste trabalho é utilizar essa mesma estrutura como entrada para a ferramenta Condado [13] e [17] na geração dos casos de teste para esses protocolos de comunicação. Uma dificuldade importante que surge na geração de testes a partir do modelo de Máquinas Finitas de Estados está na interdependência entre o aspecto controle (representado pela MFE) já implementado na PerformCharts, e o aspecto dados. Portanto, o trabalho incluirá adaptações desse modelo já implementado voltado à modelagem de desempenho para um modelo de geração de casos de teste implementando Máquina Finita de Estados Estendida. O artigo está organizado da seguinte maneira: a Seção 2 trata de aspectos relacionados a protocolos de comunicação. A Seção 3 descreve resumidamente a técnica Statecharts. A Seção 4 trata da relevância da realização de testes de protocolos de comunicação. A Seção 5 faz uma apresentação suscinta da ferramenta de geração de Testes Condado. A Seção 6 comenta o plano de trabalho. Finalmente, a Seção 7 comenta algumas conclusões além de alguns pontos para futuros trabalhos.

2. Protocolos de Comunicação

Protocolos de comunicação compõem as regras que governam a troca ordenada de mensagens entre diferentes entidades que se comunicam. Protocolos de Comunicação possuem características complexas como, por exemplo: distribuição, comunicação e sincronização. Os usuários interagem com o serviço de comunicação através de interações, chamadas primitivas de serviço, trocadas entre os chamados pontos de acesso de serviço (SAP – *Service Access Points*). À medida em que vão sendo desenvolvidos, os protocolos devem ser descritos por diversas razões: primeiramente fornecem uma referência comum entre os projetistas das diferentes partes do sistema do protocolo. O projeto deve ser checado para garantir que está logicamente correto. Em seguida, o protocolo deve ser implementado, e se está sendo amplamente utilizado, as diferentes implementações devem obedecer um padrão. O uso de linguagem natural nas especificações frequentemente contém ambiguidades além de ser difícil checar até que ponto as especificações se encontram completas e corretas. Existem diversos formalismos para se especificar protocolos de comunicação, dentre eles: Máquina Finita de Estados, Redes de Petri e Gramáticas Formais. Três técnicas de descrição formal também foram propostas com esse fim pela ISO e ITU-T (antiga CCITT): SDL, Estelle e Lotos. As duas primeiras são baseadas em extensões de FSM, permitindo com isso tratar não só o aspecto controle (relativo a ordem temporal das interações) como também o aspecto dados (referente aos parâmetros das interações, valores de variáveis, entre outros). A mesma especificação, formal ou não, pode levar a diferentes implementações de um mesmo protocolo. Por essa razão, é importante a realização dos testes de conformidade, os quais têm por objetivo verificar se uma implementação de um protocolo está conforme a sua especificação [16].

3. Técnica de Especificação de Sistemas Complexos Statecharts

Statecharts é uma técnica de descrição gráfica para a especificação comportamental baseada em estados de sistemas reativos [5], [7], [8], [9], [10]. Statecharts acrescenta às Máquinas Finitas de Estados (FSM) conceitos de decomposição hierárquica de estados (profundidade / abstração), ortogonalidade (representação de atividades paralelas) e interdependência (comunicação *broadcast*). Os elementos básicos de Statecharts para representar um sistema são: configuração (estados básicos ativos de cada componente ortogonal), eventos (externos – explícitamente estimulados e internos – automáticamente estimulados pela lógica interna de Statecharts), condição, ação, transição, expressões e variáveis. Sua notação geral:
event[condition]/action.
Action pode ser uma mudança em expressões, variáveis ou eventos disparados em outros componentes ortogonais.
Para aplicações de geração de testes, a interpretação de ação deve ser estendida também para saídas observáveis. Para fins ilustrativos, a seguir é apresentado o diagrama de estados (Figura 1) e o Statecharts equivalente (Figura 2) de um exemplo de sistema reativo [3].
Considere um sistema de semáforo. As luzes podem assumir 5 estados: *Off* (desligado), *Red* (luz vermelha acesa), *Green* (luz verde acesa), *Yellow* (luz amarela acesa), e *FlashingRed* (luz vermelha piscando). Os eventos: *RedOn*, *YellowOn*, *GreenOn*, e *FlashRedOn*, tornam seus correspondentes estados ativos. O evento *PowerOn* liga o sistema mas não acende luzes. O sistema realiza um teste automático e, caso não encontre falhas, a condição *NoFault* se torna verdadeira. Quando ocorre um evento *Reset* e a condição *NoFault* é verdadeira, o evento *RedOn* é disparado. No caso da ocorrência de falhas em qualquer estado, o evento *Fault* é ativado e o sistema retorna ao estado *Off*. O modelo especifica de uma maneira repetitiva a sequência de cores : *green*, *yellow* e *red*.

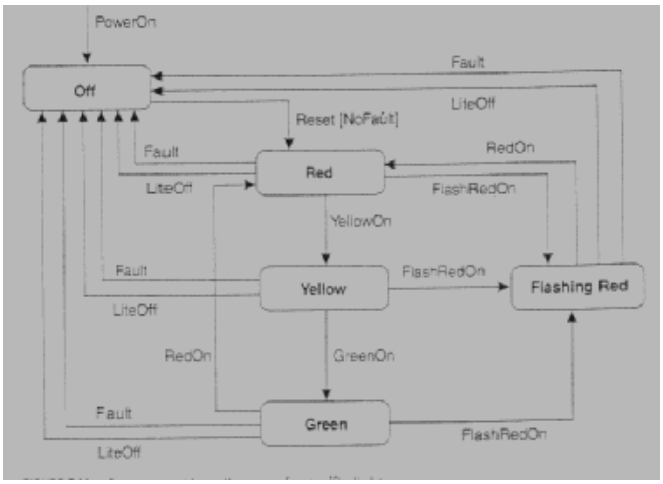

Figura 1. Diagrama de estados de um sistema de semáforo

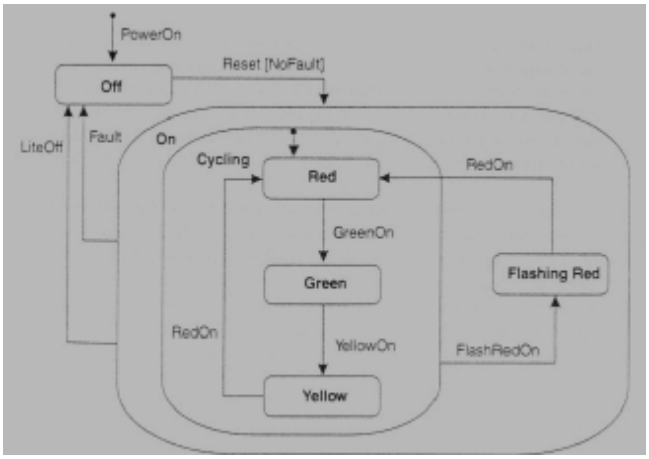

Figura 2. Representação em statecharts de um sistema de semáforo

Os superestados *On* e *Cycling* da Figura 2 mostram como as seguintes transições se tornaram simplificadas:

- Superestado *On* representa que o sistema pode estar nos estados *Cycling* ou *FlashingRed*.
- O superestado *Cycling* agrupa os estados *Red*, *Yellow*, e *Green* porque eles compartilham as mesmas transições.
- *Off* é o estado inicial, ativado pelo evento *PowerOn*.
- O evento *FlashRedOn* dispara as transições *Red- FlashinRed*, *Ÿellow- FlashingRed*, ou *Green – FlashingRed*, dependendo do estado que está ativo.
- O evento *Reset* dispara a transição *Off-On* mas, somente se a condição *NoFaults* for verdadeira. Esta é uma transição com condição de guarda.
- O evento *Reset* dispara a transição *Off- Red* pois o estado *Red* é o estado *default* para ambos os superestados *On* e *Cycling*.
- A transição sem rótulo dentro do estado *Cycling* mostra que o seu estado *default* é *Red*.

Como mencionado anteriormente, existe uma ferramenta, chamada PerformCharts, para cálculo de medidas de desempenho de um sistema reativo a partir de sua especificação em Statecharts. A interface dessa ferramenta é um programa principal na linguagem de programaçâo C++. Para facilitar o uso da PerformCharts, foi criada uma linguagem de marcação para Statecharts, baseada na technologia XML (eXtensible Markupo Language) [14] e [15], chamada PcML (PerformCharts Markup Language) e, também, uma ferramenta que a partir do sistema especificado em PcML gera o programa principal de interface com a PerformCharts, na linguagem de programação C++.

4. Testes de Conformidade

Nos últimos dez anos, foram realizadas pesquisas intensivas na área de testes de conformidade de protocolos a fim de se obter métodos melhores e sistematizados. Nos testes de conformidade, as implementações de protocolos são testadas para verificar se atendem à especificação do protocolo, assegurando compatibilidade com outras implementações. Podem também ser testadas outras propriedades da implementação não contidas na especificação do protocolo, como características específicas da implementação, robustez no tratamento de êrros e desempenho. Para certos padrões de protocolo, os comitês de padronização, têm também definido *test suites* de conformidade padronizados, que normalmente consistem num grande número de *test cases*. A execução bem sucedida destes *test cases* garantiria que as implementações testadas seguem todas as regras definidas pelo protocolo. Testes de protocolos são, principalmente, baseados na abordagem caixa-preta, também chamado teste funcional, que verifica se a funcionalidade do código está correta observando apenas as respostas do sistema sem considerar a sua estrutura interna. Este fato implica que uma especificação formal de referência deve ser fornecida como base de derivação dos casos de teste e para a análise dos resultados dos testes. Modelos largamente utilizados na literatura referente a testes de sistemas reativos, incluem Máquinas Finitas de Estado (FSM), principalmente em testes de conformidade de protocolos e de hardware[11]. Testes baseados em Máquinas Finitas de Estados têm como foco principal a geração de testes orientados a fluxo de controle tais como: "Transition Tour" (TT-Method), "unique-input-output sequence" (UIO method), "distinguishing sequence" (DS method), e

characterizing sequence (W-method) [4, 12]. Em testes de conformidade de protocolos, esses métodos têm sido exclusivamente aplicados à técnicas formais de descrição como: SDL e Estelle, e inúmeras ferramentas automatizadas têm sido desenvolvidas [6]. EFSM (Extended Finite State Machine) estendem as FSM através da inclusão de variáveis de contexto, predicados e parâmetros de interações.
Nesse modelo, uma transição é caracterizada, além dos estados de origem / destino e interação (ou evento) de entrada, também por predicados e ações. Para ocorrer a transição, é necessário o evento para dispará-la e a satisfação do predicado. Ao ser disparada a transição, a ação associada é executada, produzindo uma interação (ou evento) de saída, podendo também afetar as variáveis de entrada. As entradas são as primitivas de serviço abstratas (ASPs) e unidades de dados de protocolos (PDUs) recebidas pela IUT (implementação em teste). Os predicados são expressões lógicas envolvendo variáveis de contexto e/ou parâmetros das interações; e as ações definem as saídas produzidas e os respectivos paraâmetros, podendo também atualizar valores de variáveis. A dificuldade no uso desse tipo de modelo na geração de testes está na interdependência entre o aspecto controle (representado pelas FSM) e o aspecto dados (representado pelos predicados e ações), pois nos testes tipo caixa preta, não se tem controle sobre os dados internos à implementação, consequentemente não se sabe o valor de uma dada variável em um determinado estado, assim, se a escolha da próxima transição depender desse valor, não será possível determinar, antes da execução do protocolo, qual será o próximo estado a ser executado pela implementação [17].

5. Ferramenta Condado

Condado é uma ferramenta de geração de casos de teste tipo caixa-preta que considera tanto aspectos de controle (comportamento) como de dados / variáveis cujos valores podem causar diferentes saídas no sistema sendo testado. Os *test cases* de controle são gerados utilizando-se o método chamado T-method, ou seja, baseiam-se em subtours que iniciam e terminam no estado inicial. Do ponto de vista de teste do tipo caixa-preta, evento a ser estimulado corresponde à entrada e ação à saída observada durante o teste do sistema. A saída especificada na transição deverá ser comparada com a saída real do sistema para o teste de conformidade. A versão do sistema, já implementada para modelagem de desempenho, inclue somente ações como eventos imediatos e condições do tipo *InState* verdadeira caso o estado especificado esteja ativo. Para a geração de casos de teste, deverão ser implementadas outros tipos de condições e ações que incluam saídas observáveis. Como entrada para a Condado, a saída do primeiro módulo, ou seja, o diagrama de estados, deverá ser reescrita em PSL (*Protocol Specification Language*).

O método para geração da sequência de teste implementado pela ferramenta CONDADO é composto dos seguintes passos[20]:

1. Inicialmente são levantados os requisitos do protocolo, os quais podem estar especificados em EFSM;
2. Esses requisitos são reescritos em Linguagem para especificacao de Protocolo (PSL);
3. A partir dessa especificação em PSL, é criado um programa em Prolog chamado de gerador e, finalmente;
4. O gerador é executado para obter os casos de teste englobando dados e controle.

O passo inicial de especificação do protocolo em PSL é feito pelo usuário. Após essa especificação, a ferramenta executa os próximos passos automaticamente. Uma EFSM pode ser definida como uma Máquina Finita de Estados (FSM) com variáveis de contexto, predicados e ações. A EFSM deve estar normalizada, ou seja, as ações não podem conter instruções de desvio (if, case) e nem laços (for, while, repeat) [17].

5. Plano de Trabalho

A proposta para a geração automática de casos de teste de conformidade para software de aplicações em protocolos de comunicação, baseia-se em adaptações necessárias a serem realizadas no código da ferramenta PerformCharts existente no LAC, que gera medidas de desempenho.
Tais adaptações incluem:

1. Inclusão do elemento "Variable" já existente no formalismo Statecharts mas não implementada dentro da PerformCharts.
2. Na PerformCharts foi implementado o elemento "Action" onde seria um evento primitivo sem nenhuma taxa estocástica associada. Para geração de casos de testes, este elemento poderia simplesmente ser uma saída (de um string ou valor) para poder verificar a coerência após a reação a um evento.
3. Na PerformCharts foram implementados alguns tipos de “Conditions” já definidos no formalismo Statecharts. É preciso estender a implementação para incluir os outros tipos ainda não implementados que utilizam variáveis (elemento "Variable").
4. Atualmente, a ferramenta que gera os casos de testes assume que a EFSM esteja numa forma que poderá ser interpretada por um programa PROLOG. Mas a idéia é que esta ferramenta possa utilizar uma interface baseada em XML. Então, um dos trabalhos seria, baseada na experiência em PcML[1], pesquisar a existência de uma linguagem baseada em XML para especificar EFSM. Esta abordagem necessita que se mude o método de leitura da especificação dentro da ferramenta Condado.

É preciso lembrar que dependendo da complexidade do sistema especificado, poderá haver problema de explosão de estados pois a conversão em si da especificação Statecharts

para uma EFSM não consegue resolver este problema. Os estudos de como evitar que haja a tal explosão de estados não é escopo do presente trabalho.

7. Conclusões

Protocolos de comunicação, sendo um tipo de sistema reativo, possuem peculiaridades difíceis de serem representadas em sua especificação. Linguagem natural não é uma forma recomendada para descrever tais sistemas pois podem conter ambiguidades e incorreções além de não descrever a especificação de forma completa. Portanto, o uso de formalismo é indicado para representá-los. Statecharts possuem recursos que facilitam a representação de tais sistemas na maioria dos casos. Existe uma ferramenta já desenvolvida no LAC/INPE, para avaliação de desempenho, chamada PerformCharts [21] e [22], que associa a representação do sistema em Statecharts a uma solução matemática, em particular a cadeia de Markov, gerando como saída diagrama de estados da máquina finita de estados. A proposta deste trabalho é, a partir desse sistema já funcional, implementar as devidas adequações para geração de casos de teste, gerando como saída diagramas de estado da máquina finita de estados estendida a ser usada como entrada para a ferramenta Condado. Esta, engloba de forma unificada tanto aspectos de controle quanto de dados. Seu principal potencial de aplicação, seria a validação de sistemas de protocolos de comunicação de aplicações espaciais.

7. Referências

[1] Amaral, A.S.M.S.; Veloso R.R. & Vijaykumar, N.L., "PcML Reference Manual", *Relatório Técnico a ser publicado*, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brazil.

[2] Ambrósio, A.M., "Geração Automática de Casos de Teste do Comportamento de Sistemas de Software que se Comunicam", *Proposta de Doutorado (CAP-INPE)*, São José dos Campos, SP, Brazil, 2002.

[3] Binder, R.V., *TESTING OBJECT-ORIENTED SYSTEMS Models, Patterns, And Tools*, Addison-Wesley, USA, 2001.

[4] Bochman, G. & Petrenko, A., "Protocol Testing: Review of Methods and Relevance for Software Testing", *Proceedings of the 1994 International Synposium on Software Testing and analysis*, Seatle, Washington, USA, 1994, 109-124.

[5] Drusinsky, D. & Harel, D., "Using Statecharts for Hardware Description and Synthesis", *IEEE Transactions on Computer-Aided Design*, 8(7), 798-807.

[6] Dssouli, H.; Salek, K.; Aboulhamid, E.; En-Nouaary, A. & Bourhfir, C.; "Test Development for Communication Protocols: Towards Automation". *Computer Networks*, 1999, 1835-1872.

[7] Harel, D., "Statecharts: a visual formalism for complex systems", *Science of Computer Programming* , 1987, 8, 231-274.

[8] Harel, D. & Naamad, A., "The STATEMATE Semantics of Statecharts", *ACM Transactions on Software Engineering, 1996*, 5(4), 293-333.

[9] Harel, D.; Pnueli; A., Schmidt, J.,& Sherman, R., "On formal semantics of Statecharts", *IEEE Symposium on Logic in Computer Science*, Ithaca, USA., 1987.

[10] Harel, D. & Politi, M., *Modeling Reactive Systems with Statecharts: the Statemate Approach*, McGraw-Hill, USA, 1998.

[11] Hong, H.S.; Lee, I.; Sokolsky, º & Cha, S.D., "Automatic Test Generation from Statecharts Using Model Checking", *Proceedings of the First Workshop on Formal Approaches to Testing of Software (FATES 01),* Aalborg, Denmark, 2001, 15-30.

[12] Lee, D. & Yannakakis, M., "Testing Finite-State Machines: State Identification and Verification", *IEEE Transactions on Computers*, 1994, 43, 3.

[13] Martins, E.; Sabião, S.B. & Ambrósio, A.M., "ConData: a tool for automating Specification-based Test Case Generation for Communicating Systems. *Software Quality Journal*, 8(4), 303-319, 1999.

[14] Maruyama, H., *XML and Java – Developing Web Application.*, Addison Wesley, Ca, 2002.

[15] Ray, E. T. , *Perl and XML*, O´Reilly & Associates, Inc., USA, 2002.

[16] Rayner, D., "OSI Conforming Testing", *Computer Networks and ISDN Systems*, 1987, 14, 89-98.

[17] Sabião, S.B., "Um Método para Geração de Testes Baseado em Máquina Finita de Estado Estendida Combinando Técnicas de Teste Caixa Preta". *Dissertação de Mestrado Instituto de Computação (Unicamp),* Campinas, SP, Brazil, 1999.

[18] Sarikaya, B.; Bochmann, G.V. & Cerny, E. "A Test Design Methodology for Protocol Testing", *IEEE Transactions on Software Engineering*, 1987, SE-13, 5, 518-53.

[19] Sarikaya, B.; Bochmann, G.V. & Cerny, E. "A Test Design Methodology for Protocol Testing", *IEEE Transactions on Software Engineering*, 1987, SE-13, 5, 518-53.

[20] Ural, H. & Probert , R.L., "User_guided Test Sequence Generation", *Protocol Specification, Testing and Verification III*, 1983, 421-436.

[21] Vijaykumar, N.L., "Statecharts: Their use in specifying and dealing with Performance Models", *Ph.D. Thesis Aeronautical*

Institute of Technology (ITA), 1999, São José dos Campos, SP, Brazil.

[22] Vijaykumar, N. L.; Carvalho & S.V., Abdurahiman, V., "On proposing Statecharts to specify Performance Models", *International Transactions in Operational Research*, 2002, 9(3), 321-336.